SKROLLI 2015.4

SKROLLI

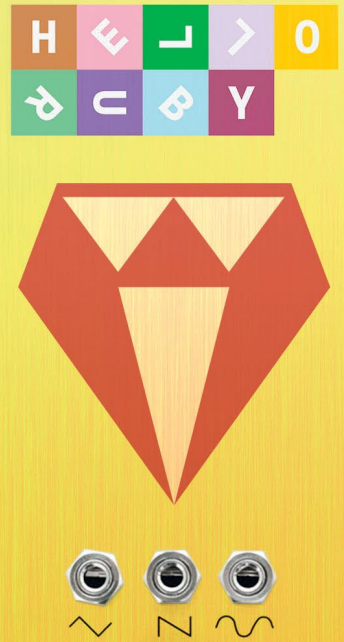
2015.4
SKROLLI.FI

Tietokonekulttuurin erikoislehti

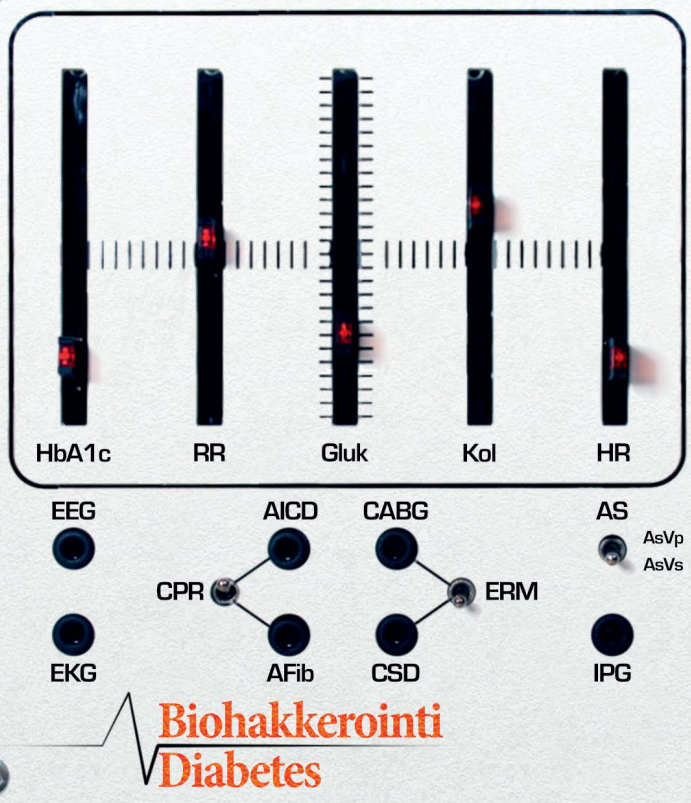
Neuvostotasavallan tietokoneet

Pelaamalla koodaajaksi

koodisatuja



UUSI PILVIKONE Solu



Biohakkerointi Diabetes

C++ **BASIC** **SAILFISH**

3 **Pääkirjoitus**

4 **Kirja-arvostelu: Hello Ruby**
Lasten koodisatukirja kasvattaa uutta sukupolvea.

5 **Mikrokivikausi**

6 **Unelma kaikesta pilvessä**
Taskutietokone Solu lupaa jotain aivan uutta.

8 **Pelaamalla koodaajaksi**
Opettele ohjelmoinnin perusteet pelien parissa.

12 **Hitler-videomeemien anatomia**
Diktaattoria on pilkattu jo lähemmäs vuosisadan verran.

16 **Armottomat pikselit**
Villin lännen seikkailut ovat aina kiehtoneet pelintekijöitä.

20 **Teknologia diabeetikon apuna**
Softa, rauta ja avoin lähdekoodi mullistavat diabeteksen hoitoa.

24 **Internetin hallintomallit**
Onko netissä oikeuksia vai vain mahdollisuuksia?

29 **Kolumni: Janne Sirén**
Omaiskoodaaja voi auttaa kroonisten sairauksien itsehoidossa.

30 **C++ standardimankelissa**
Näin ohjelmointikielen yhtenäisyys varmistetaan.

33 **Postipalsta**

34 **Turku – Suomen syntetisaattoripääkaupunki**
Synamiitti kokoaa vuosittain yhteen innokkaimmat harrastajat.

37 **Äänisynteesi**
Näin ykköset ja nollat muovautuvat musiikiksi.

44 **Jollalla perille**
Sailfish-ohjelmointisarjan kolmannessa osassa otetaan käyttöön kartat ja satelliittipaikannus.

52 **Tietoa kantaan ja takaisin**
Tällaisilla tekniikoilla tiedonpalaset pysyvät järjestyksessä.

56 **Kiintolevy Commodore 64:ään**
Tähän ei 1541 pysty – isotkin ohjelmat latautuvat sekunneissa.

60 **Nyky-Basicit**
Kun Basic on selkäytimessä, sitä haluaa käyttää nykyajankin hommissa.

62 **ZX Spectrum Vega**
Saarivaltion suosikin emuloitu eläke.

64 **ZX Evolution**
Kuusnepaa siellä saatu ei, joten Spectrum idässä voiton vei.

68 **BK-0011**
Kymmenvuotinen kaipuu päättyi – käsissä on viimein neukkukone.

72 **Pelejä tuunaamassa, osa 1**
Mitä tapahtuu, kun pelimaailman rakenteet avataan sorkittavaksi?

78 **Pelihistoriallinen outolintu**
Metal Gear -sarjan mustan käärmeen tarina.

82 **Ei näin!**
Imagine Software oli räiskyvä tähdenlento, joka sammui nopeasti.

Ruby-tyttö seikkailee Linda Liukkaan Hello Ruby -kirjassa, josta on arvio sivulla 4.

Ville-Matias Heikkilä
päätoimittaja

skrolli.int

Nyt se on päätetty: Skrolli levittäytyy ulkomaille.

Skrolli aloittelee joukkorahoituskampanjaa, jonka kautta on tarkoitus saada aikaan ainakin yksi englanninkielinen numero. Ensimmäisen lehden sisältö olisi vielä käännöksiä suomenkielisen lehden parhaimmistosta, mutta myöhemmin kieliversioiden olisi tarkoitus tukea toisiaan. Mikäli ulkomaantilauksia tulee riittävästi, voisimme vihdoinkin ruveta tekemään Skrollia ihan oikeasti työksemme, mikä varmasti myös parantaisi suomenkielistä versiota entisestään.

Skrollin kaltaista lehteä ei suuressa maailmassa oikeastaan ole. Suurin osa perinteisistä tietotekniikan yleislehdistä on joko lopettanut tai kärsii talousvaikeuksista. Jotkin erikoistuneemmat lehdet pärjäävät vielä – esimerkiksi Linux Format, monet peli- ja retroilulehdet ja toki myös tiedejulkaisut. Kahden numeron verran ilmestynyt Sceen oli yleishengeltään varsin lähellä Skrollia, joskin se keskittyi aika vahvasti demoskeneen ja muihin luoviin tietokonealakulttuureihin.

Olemme siis muihin verrattuna varsin laaja-alainen lehti: yhdistämme monia alakulttuureja ja kuppikuntia. Tällainen pirstaloitumattomuus on muutenkin tyypillistä suomalaiselle tietokonekulttuurille. Meillä erityyppiset hakkerit, harrastajat, startup-yrittäjät ja pelaajat ovat paljon lähempänä toisiaan kuin isommissa maissa. Kansakunnan pienuudella on varmasti tekemistä asian kanssa, mutta niin myös Assemblyn ja Alternative Partyn kaltaisilla tapahtumilla. Nämä ovat haalineet laajalti eri ”skenejä” saman katon alle ottamaan kontakteja toisiinsa.

Meidän ei ole myöskään tarvinnut ajatella kaikkea ensisijaisesti rahan kautta – elämmehän vielä ainakin toistaiseksi modernissa hyvinvointivaltiossa. Näin ihmiset ovat voineet hyvin omistautua erilaisille intohimoisille harrasteprojekteille, kuten erilaisille avoimen lähdekoodin ohjelmille tai vaikkapa Skrolli-lehdelle. Suomalaisilla on siis edelleen suhteellisen vahva käsitys siitä, mitä tietokoneharrastus voi olla.

Suomalaisessa tietokonekulttuurissa on aina ollut paitsi hyvä yhteishenki, myös vahvat kansainväliset yhteydet. Ja niitä tarvitaan nyt, kun yritämme levittää Skrollin ilosanomaa muiden maiden tietokonekulttuuripiireihin. Oli viiteryhmäsi siis mikä hyvänsä, niin kerro Skrollista ja sen englanninkielisen version joukkorahoituskampanjasta eteenpäin! Mitä laajemmin erilaisilta tietokonekulttuurin osa-alueilta saamme kansainvälistä huomiota, sitä paremman ja monipuolisemman lehden voimme saada aikaan!

skrolli.fi/international

Skrolli
Tietokonekulttuurin erikoislehti

Yhteydenotot toimitus@skrolli.fi
Ircnet: #skrolli
skrolli.fi

Päätoimittaja Ville-Matias Heikkilä
Toimituspäälliköt Toni Kuokkanen & Ninnu Koskenalho
Taiteellinen johtaja Nasu Mäki-Petäys
Kuvatoimittaja Mitol Meerna
Mediamyynti Jari Jaanto
Talous Anssi Kolehmainen

Muu toimitus Mikko Heinonen, Päivi Julin, Jukka O. Kauppinen, Ronja Koistinen, Teemu Likonen, Annika Piiroinen, Manu Pärssinen, Janne Sirén, Suvi Sivulainen

Tämän numeron avustajat Jani Halme, Mika Haulo, Chris Helenius, Pasi Huttinen, Matti Hämäläinen, Panu Kalliokoski, Jussi Kasurinen, Janne Katajamäki, Jarkko Lehti, Tapio Lehtimäki, Miikka Lehtonen, Sakari Leppä, Asser Lähdemäki, Visa-Valtteri Pimiä, Markku Reunanen, Petri Saarikoski, Mikko O. Torvinen, Erno Vanhala, Ville Voutilainen, Wallu, Antti Ylikoski

Julkaisija Skrolli ry

Painopaikka Tammerprint, Tampere,
ISSN 2323-8992 (painettu)
ISSN 2323-900X (verkkojulkaisu)

Etukannen kuva:
Nasu Mäki-Petäys

Sisällysluettelon kuva:
Hello Ruby, Otava

YMPÄRISTÖMERKKI
MILJÖMÄRKT
441 878
Painotuote

Hello Ruby

Mediassa paljon näkyneen Linda Liukkaan ”koodisatukirja” Hello Ruby ilmestyi syksyllä suomeksi ja englanniksi. Kirja on söpö, mutta pääseekö se tavoitteeseensa?

Teksti: Ronja Koistinen

Kuvat: Otso Kaijaluoto, Maija Tammi (Otava)

Viime vuosina on intoiltu paremman, aiempaa nuoremmille suunnatun ja paremmin saavutettavan tietotekniikkakoulutuksen perään. Ohjelmistoalan veteraanit ovat tippa linssissä muistelleet lapsuutensa 80-lukua ja harmitelleet, ettei nykyään ole tenavilla yhtä matala kynnys tarttua koneeseen ja alkaa luoda sillä uutta.

Vastaukseksi toiveisiin on suunniteltu ohjelmointiopetusta alakouluihin, ja julkaistu kirjallisuutta. Suomessa aihepiirin tunnetuimpia puuhaajia on ollut Linda Liukas, jolta ilmestyi tänä syksynä lasten ohjelmointikirja Hello Ruby. Kirja syntyi joukkorahoitusprojektina Kickstarterissa, jossa se menestyi mallikkaasti.

Aiemmin Liukas on tunnettu muun muassa Amerikassa pyörittämästään Rails Girls -projektista, joka on keskittynyt ohjelmoinnin opettamiseen tytöille ja naisille. Suomessa hän on saanut vuonna 2014 lastenkulttuurin valtionpalkinnon tulevaisuuteen suuntautuneen lastenkulttuurin kehittämisestä. Vuonna 2015 hänet valittiin vuoden tietotekniikkavaikuttajaksi Suomessa.

Liukas kirjoitti kirjan aluksi englanniksi. Teoksen julkaisija Suomessa on Otava ja suomentaja Kirsikka Myllyrinne.

Koodilistauksia ei tarvitse tavailla

Hello Ruby ei oikeastaan ole ohjelmointikirja, vaan kutsuu itseään koodisatukirjaksi. Se on tyylilajiltaan lastenkirja värikkäine kuvituksineen ja hassuine tarinoineen. Samalla se pyrkii virittämään lasta loogiseen ajatteluun, päättelemiseen ja ongelmien pilkkomiseen. Esipuhe käyttää ilmausta ”ohjelmoinnillinen ajattelu”.

Kirjassa on kaksi osaa. Alkupuoli on satu Ruby-nimisestä tytöstä ja hänen seikkailuistaan vekkulien hahmojen kanssa. Seikkailussa on erilaisia vaiheita, esteitä ja ongelmia, joita Ruby ja hänen ystävänsä ratkovat päättelemällä, suunnittelemalla ja rakentelemalla.

Satuosan jälkeen seuraa tehtäväkirja. Siinä on erilaisia kuvitettuja pulmia lapsen ratkottavaksi. Ne ovat kuviopäättelyitä, sääntöjen tulkitsemista, Venn-diagrammeja ja jopa loogisia totuusarvoja. Aihepiireinä ovat värikkäät ja arkiset lasten asiat: lelut, nuket, eläimet ja leivonnaiset. Mitään ohjelmointikieltä kirja ei opeta. Ongelmi-

en tarkastelu tähtää ohjelmoinnillisen päättelemisen harjoitteluun.

Paketti saattaa olla liian tuhti

En ole varhaiskasvatuksen enkä lastenkirjojen asiantuntija, mutta minua hieman huolettaa, että Hello Rubyn satu- ja tehtäväosuus saattavat olla epätasapainossa. Kirja sanoo olevansa tarkoitettu toisaalta alle kouluikäisille, toisaalta takakannen mukaan ”kaikille yli 5-vuotiaille”. Kuusivuotiaalle Hello Rubyn satu saattaa olla epäkiinnostava ja juoneton sarja sattumuksia. Tehtäväkirjan pulmat taas saattavat olla liian tylsiä, abstrakteja ja vaikeita kuusivuotiaalle ja sopisivat pikemminkin 8–10-vuotiaalle.

Kirjassa on myös muutamia kohtia, jotka aiheuttavat pikkumaista kulmien kurtistelua koodaukseen virittyneessä lukijassa. Ensimmäisenä silmään osuu jo mainitsemani takakannen teksti ”kaikille yli 5-vuotiaille”. Tarkoittaako se ikä ≥ 5 vai ikä > 5? Riittääkö 5½-vuotias, vai pitääkö olla täyttänyt kuusi?

Toinen, äskeistä huolestuttavampi huolimattomuus esiintyy tehtäväosiossa sivulla 108. Aukeaman täyttää nop-papeli, jossa edetään radalla. Yhdessä kohdassa on ohjeet, mitä tehdään, jos heittää nopalla silmäluvun alle kolme tai enemmän kuin kolme. Sen sijaan mitään ohjetta ei anneta siltä varalta, että heittää tasan kolme.

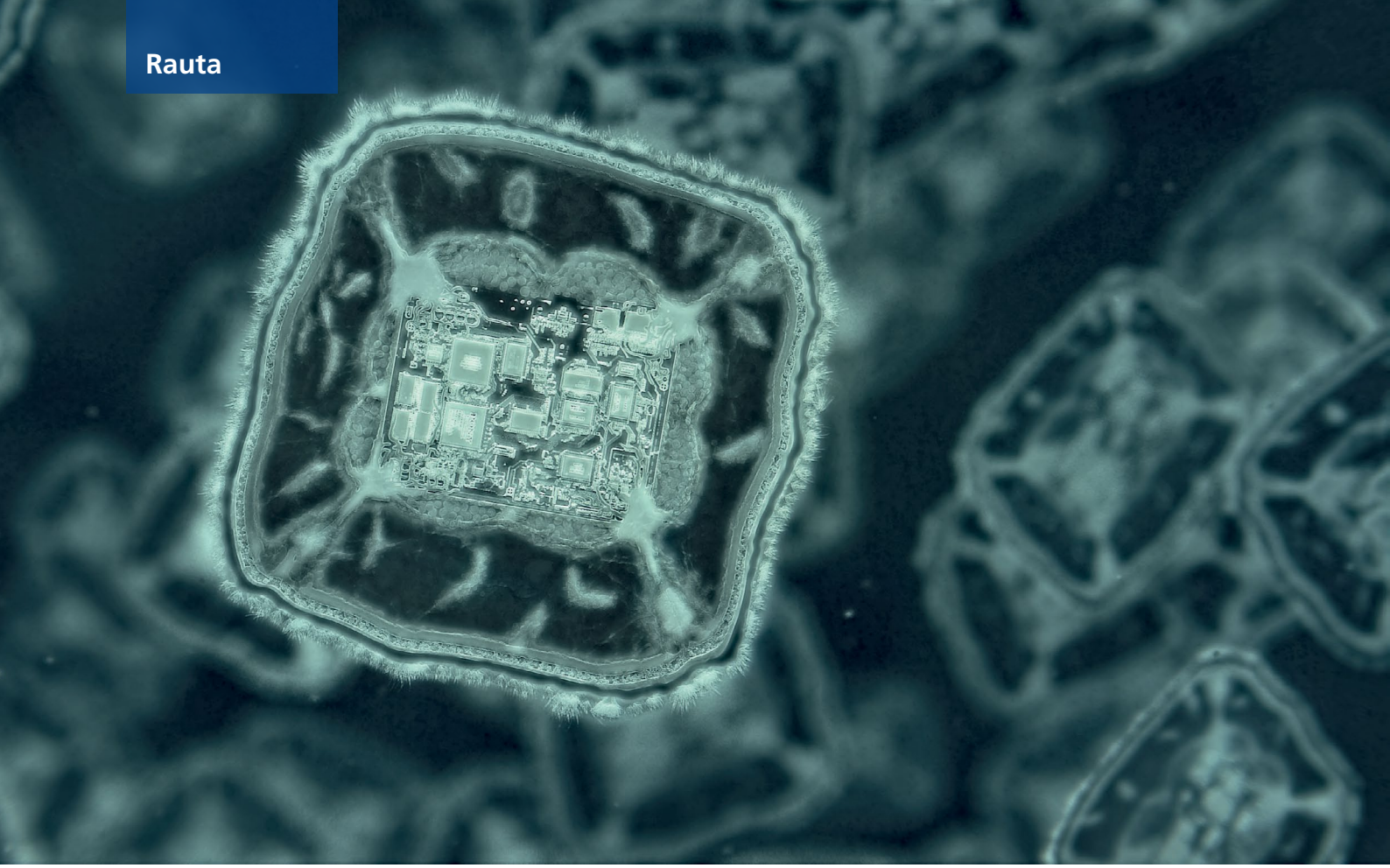
Nämä pikkujutut saattavat toki olla myös suomennoksen mokia.

Tätä lisää!

Nipotuksestani huolimatta Hello Ruby on todella vahva avaus 2010-luvun nopeasti käyntiin lähteneelle ohjelmointipedagogiikalle. Sekä koulutuspolitiikan että IT-yritysmaailman setämiehet puhuvat paljon siitä, kuinka ohjelmointi on kaikkien alojen tulevaisuus – vain sillä voi automatisoituvassa maailmassa turvata elantonsa. Opettajat koulumaailman kentällä ovat kuitenkin pian helisemässä, jos ei suuria puheita saada lunastettua toimiviksi menetelmiksi.

Tarvitaan paljon lisää Hello Rubyn, Linda Liukkaan toisen projektin Rails Girlsin ja hacklabien tapaisia konkreettisia tekoja ja ennakkoluulottomia ympäristöjä. Sitten varmasti pikkuhiljaa löydetään toimivat tavat saada nuori polvi oikeasti koodaamaan.

Hello Ruby. Maailman paras koodisatukirja!
Linda Liukas, Otava 2015.
Alkuteos: *Hello Ruby: Adventures in Coding.*
Suomentanut: Kirsikka Myllyrinne.
ISBN: 978-951-1-29038-4
Kovat kannet, 111 sivua



Solu
Unelma kaikesta pilvessä

Lokakuussa suomalainen Solu keikutti IT-maailman aaltoja puusta valmistetulla taskutietokoneella, joka lupaa viedä kaiken pilveen. Skrolli kurkisti viilun ja hypen taakse.

Teksti: Janne Sirén, Visa-Valtteri Pimiä Kuvat: Sakari Leppä, Solu Machines

Kaikki alkoi musiikkilevyistä. Digitaalinen pilvivallankumous, nimittäin. Ensiksi musiikki samplattiin analogisen LP-levyn sijaan digitaaliselle CD-levylle, sitten CD-levyt ripattiin tietokoneen kiintolevylle. Kolmannessa vaiheessa digilevyt ladattiin suoraan internetistä kiintolevylle. Lopulta joku keksi että miksi ladata, kun voi striimata – ja kun voi striimata, voi musiikkia myös jakaa yhtä vaivattomasti.

Kesti vuosikymmeniä, että teknologia, lisenssit ja liiketoimintamallit kypsyivät idealle. Taylor Swiftin ja parin rokkikukon kipuilua lukuun ottamatta musiikissa ollaan maalissa, sitä on saatavilla internetistä mistä vaan, milloin vaan ja vieläpä laillisesti. Elokuvat, TV-sarjat, tieto ovat nekin pitkälti jo verkossa.

Solu yrittää saada pilveen loputkin – raudan ja softan.

Historian läksyjä

Laitetta pilven ympärille on toki rakennettu aikaisemminkin. Microsoft joutui myrskyn silmään Xbox One -pelikonsolillaan, jonka ensi-iteraatiossa omistamasi sisällön piti olla käytettävissä millä tahansa Xbox Onella. Kinect-kamera tunnistaisi naamasi, verkkoyhteys siirtäisi sisällön ja varmistaisi käyttöoikeudet. Vaatimus verkkoyhteydestä kuitenkin närästi liikaa ydinsukellusvenemiehistöjä, ja markkinapaineessa Microsoft vesitti osan suunnitelmistaan.

Solu on pieni, orgaaninen tabletti: Lasia, puuta ja sisällön mukana elävä pilvikäyttöliittymä, SoluOS.

Tietokoneiden puolella Steamin kaltaiset pelikaupat ja monet ammattisovellukset, kuten Adobe Creative Suite ja Microsoft Office, ovat jo pitkälti siirtyneet pilveen. Ohjelmia voi käyttää missä ja milloin vain, kunhan verkkoyhteys varmistaa käyttöoikeuden jatkumisen. Lisenssistä maksetaan kuitenkin kuukausittain, mistä kaikki eivät tykkää. Lisäksi tämä yleensä tarkoittaa sovellusten erillistä asentamista eri koneille tai kevyempiä web-versioita. Jonkin verran ohjelmistokomponenttejakin jo striimataan.

Sisällön automaattivarmennus pilveen on myös yleistynyt. Sisältö ja joskus jopa sovellukset voidaan tarvittaessa palauttaa tai siirtää toisaalle. Yksi ohjelmisto, tietokoneilla tai mobiilissa, ei kuitenkaan ole vielä pilvisovellus: käyttöjärjestelmä. Niinpä viimeinen niitti on puuttunut. Laitteet ovat edelleen omia saarekkeitaan, joille raken-

netaan paikallinen infrastruktuuri ja vasta sitten tuodaan tuontisisältöä. Aina kun istahdat uudelle tietokoneelle, joudut enemmän tai vähemmän aloittamaan alusta.

Rajaa hämärtämässä

Kysyimme yhdeltä Solun perustajista, toimitusjohtaja Kristoffer ”Setok” Lawsonilta, mitä he yrittävät tehdä eri tavalla kuin muut.

Kristoffer myöntää, että osansa Solussa on ihan teknologian yleisellä kehityksellä. Idea Solu-taskutietokoneesta ulottuu parin vuosikymmenen taakse, kun Kristoffer ensimmäisiä kertoja tutustui silloin jo varsin suureksi kasvaneeseen internetiin, mutta vasta nykyisin taskurauta ja langattomat verkkoyhteydet riittävät toteutukseen. Solu on modernisti kosketusnäyttöön naitettu Nvidian Tegra K1 -system-on-a-chip, päivän akkukestolla ja wlan-yhteydellä. Lisälaitteet ulkoista 4K-näyttöä myöten liittyvät siihen USB-C-portin kautta.

Toisin sanoen Solu on pieni, neliönmuotoinen tabletti, jonka voi kytkeä myös isoon näyttöön. Solu on trendikäs ja nätti kuin kuusenkerkkäkarkki – se löytyy Kickstarterista, ja pintaan on vuoltu lastu lähimetsää. Nykypäivänä näillä ominaisuuksilla tosin erottuu toisista yhtä heikosti kuin hipsteri Kånken-repulla. Ensivilkaisu mikrobiologiaa muistuttavaan käyttöliittymään on sentään selvästi poikkeava, mutta sekään ei täysin vastaa kysymykseen, mitä tällä tehdään. Esijulkaisuvaiheessa Solu on vielä vähän mysteeri.

Kristofferilla on kuitenkin kiinnostava pointti: Mitä ikinä Solulla tekeekin, sitä voi tehdä millä tahansa Solulla, koska kaikki Solussa on natiivisti pilvessä. Jos Solu tipahtaa järveen, uuteen Soluun tarvitsee vain kirjautua sisään, ja kaikki ohjelmat ja dokumentit ovat heti käytettävissä. Jos tallennustilaa uupuu, pilvessä on automaattisesti lisää. Offline-käytöstä vastaa älykäs välimuisti, joten siellä ydinsukellusveneessäkin Solu on ainakin teoriassa tallentanut kaiken tähdellisen välimuistiin ja peli jatkuu. Jos Solu toimii kuten lupaa, konsepti on kiistatta hieno.

Spotify-malli ekosysteemille

Solun, kuten minkä tahansa uuden alustan, suurin haaste ovat nykypäivän vaatimukset toimivasta ohjelmistoekosysteemistä. Siis juuri se yksi asia, jota tässä yritetään siirtää pilveen, on samalla ylipäätänsä kaikkein vaikein ratkaistava – appsit. Tähänkin Kristofferilla on vastaus, ja oli Solun mahdollisuuksista mitä mieltä tahansa, idea ainakin on erilainen: kaikki Solulle tehdyt sovellukset nimittäin kuuluvat Solun App Libraryn kuukausimaksuun.

Solu on siis valinnut Spotifyn liiketoimintamallin. Käyttäjät maksavat sovelluksistaan all you can eat -kuukausimaksua, ja sovelluskehittäjille maksetaan sovellusten käyttötilastojen mukaan. Koska sovellukset ovat näin erillismaksuttomia, niitä on tarkoitus voida jakaa käyttäjältä toiselle yhtä helposti kuin kissavideoita. Tämä liiketoimintamalli on niin tärkeä osa yrityksen ideologiaa, että Kristofferin mukaan se on peräti kirjattu manifestiin, johon sijoittajat eivät saa puuttua.

Pohjalla on kuitenkin jotain tuttuakin. Solun käyttöjärjestelmänä on muovattu Android, jonka sovelluksia voi käyttää, kunhan ei kaipaa niihin Solun erityisominaisuuksia. Oman Solu-app-ekosysteemin tarvetta ei silti voi vähätellä. Solun käyttöliittymä kun pyrkii olemaan hyvin erilainen, uudesta pilvikollaboraatio- ja liiketoimintamallista puhumattakaan. Lisäksi Solun käyttötilanteet vaihtelevat sen pieneltä neliönäytöltä suurelle tietokonenäytölle, jolloin appien tulisi osata hyödyntää pientä näyttöä kakkoskäyttöliittymänä.

Yhdeksi vauhdittajaksi toivotaan demoskenen ja hakkerimentaliteetin vaikutusta. Solu on Kristofferin mukaan jo nyt saanut siitä osansa, mikä näkyy muun muassa käyttöjärjestelmän reippaassa kehitystahdissa ja käyttöliittymän efekteissä. Lisää kontribuutioita odotellaan, kun ensimmäiset Solu-ohjelmistokehittäjät saavat kehityslaitteensa toivottavasti jouluksi.

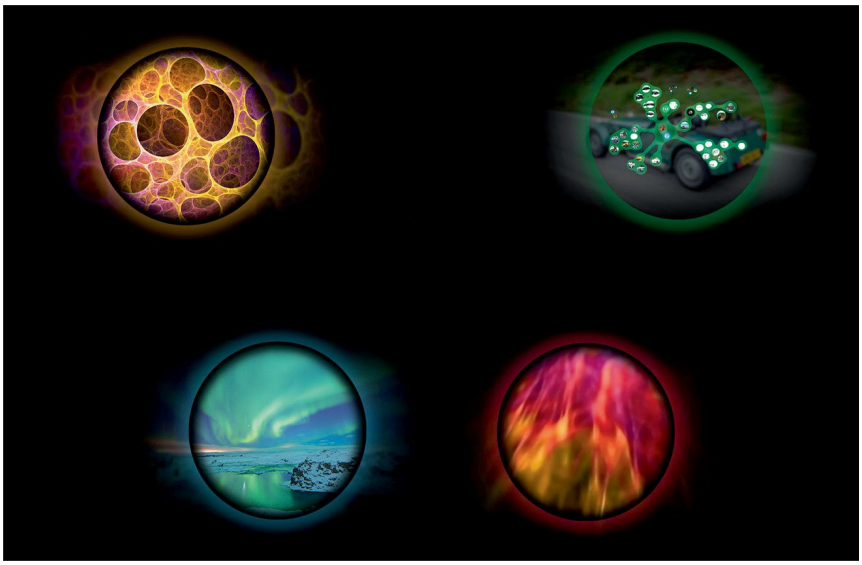
Siniristi- vai pääkallolippu?

Solut valmistetaan Suomessa ja myös niiden pilvipalvelimet sijaitsevat Suomessa, mikä on mainio asia. Haastoimme kuitenkin Kristofferia kysymällä, mitä mieltä hän on, jos raudan käyttäjät hylkäävät emopalvelut ja perustavat sen sijaan omia piraattiservereitään. Kristoffer heittää avoimen pallon mieluummin laitekehityksen suuntaan:

”Lähtökohtaisesti Solu toimii meidän palvelumme kanssa – sitä kautta saadaan kollaboraatio ja ohjelmien liiketoimintamalli pelaamaan, siinä on meidän leipä palvelun pyörittämisessä. On käyttäjien etujen mukaista, että yrityksemme toimii. Rautapuolella olemme harkinneet avointa lähdekoodia, sillä ei olisi ollenkaan haitaksi, että palveluun kiinnittyisi hyvinkin erilaisia laitteita.”

Ihan kaikki ei Solussakaan ole pilvessä: se käyttöjärjestelmä. Käyttöjärjestelmän päivitykset tehdään manuaalisesti kuten Androideissa yleensäkin. Solun tuotantoversio ilmestyy keväällä noin 450 euron hintaan (Kickstarter 350 euroa).

Lisätietoja: www.solu.co



SoluOS-käyttöjärjestelmässä elämänsä voi jakaa (yhteistyö)projekteihin.

Pelaamalla koodaajaksi

Voiko pelaamalla oppia ohjelmoimaan?

Koodaaminen on hauskaa, ja niin ovat videopelitkin. Asioilla on perinteisesti ollut se yhteys, että koodaamalla tehdään pelejä, mutta eihän sen tarvitse olla pelkästään näin. Joskus peli voi myös kertoa koodaamisesta. Skrolli testasi neljää erilaista peliä, joissa ohjelmointi on pelimekaniikkana.

Teksti: Miikka Lehtonen
Kuvat: Nasu Mäki-Petäys, Miikka Lehtonen, Jani Halme

Monissa peleissä ratkotaan tiettyjen sääntöjen rajoissa loogisia pähkinöitä, ja sitähän ohjelmointikin pohjimmiltaan on. Algoritmisena ajatteluna tunnettu käsite on kaiken ohjelmoinnin perusta. Sillä tarkoitetaan yksinkertaisesti sitä prosessia, jonka avulla abstrakteja ongelmia muutetaan loogisiksi askeleiksi, joita tietokone pystyy seuraamaan. Niinpä ei ole mitenkään hirvittävän yllättävää, että useissa peleissä on joko lainailtu ideoita ohjelmoinnista tai suoranaisesti tehty ohjelmoinnista pelimekaniikka.

Kun ryhdyin tutkimaan ideaa tarkemmin tätä artikkelia varten, kävi ilmi, että erilaisia ohjelmoinnin ympärillä pyöriviä pelejä riittää yllin kyllin, ja että rajaukset ovat usein hieman hankalia. Mitä tehdä esimerkiksi peleille, joissa suunnitellaan tekoälyrutiineita erilaisille roboteille tai vaikka liukuhihnojen varassa toimiville tuotantolaitoksille? Koska tila ja aika ovat rajallisia, ne jäivät tämän jutun ulkopuolelle.

Lopulta seulan läpäisi neljä peliä, joissa kuljetaan tietyn akselin päästä toiseen. Yhdessä päässä on selvästi viihdepeliksi suunniteltu teos, jossa ohjelmointi on mukana, mutta kivasti ja lähestyttävästi esitettynä. Toisessa päässä on pelillistetty ohjelmointikoulu, jossa jo osaavat voivat hioa taitojaan. Mitä mahtaakaan mahtua välimaastoon?

Human Resource Machine

Human Resource Machine on testin tuorein ja selvästi pelimäisin julkaisu. Taustalta löytyy yksinäinen kehittäjä, jonka edellisessä pelissä poltettiin takassa erilaisia esineitä. Siinä se. Omaperäiset ideat ovat siis tuttuja, ja sellainen on myös Human Resource Machine.

Pelaaja on suuren yhtiön tuorein

Human Resource Machinen värikäs grafiikka ja hauska huumori peittävät alleen pelin, jossa pulmat ovat oikeasti haastavia – ja koodista tulee melkoista spagettia.

työntekijä, jonka tehtävänä on suorittaa yksinkertaisia loogisia tehtäviä sen verran hyvin, ettei hänen työpaikkaansa tarvise korvata tietokoneella.

Human Resource Machine on selvästi suunniteltu ihmisille, jotka eivät ole ikinä koodanneet. Jo pelin esittelytekstissä Steam-verkkokaupan sivuilla mainostetaan, että ei ole mitään syytä olla huolissaan, vaikka ei olisi koskaan koodannut. Kaikki opetetaan kyllä pelin aikana. Ja niin myös tehdään.

Ruudun vasemmassa laidassa on inbox, johon saapuu syötteenä erilaisia numeroita ja kirjaimia, kentästä riippuen. Oikeassa laidassa on outbox, johon duunarin pitäisi saamansa numerot toimittaa tiettyjen pomon antamien sääntöjen rajoissa.

Tätä varten hänellä on käytössään lattialla olevia lokeroita, joihin numeroita voi väliaikaisesti laskea, sekä käskyjä, joilla numeroita siirrellään lokeroiden välillä, lasketaan yhteen tai vähennetään toisistaan tai vaikka kasvatetaan yhdellä. Heitetään mukaan vielä vähän loogisia ohjauskäskyjä, joilla hypätään työntekijän ohjeissa eri kohtaan sen mukaan, onko kädessä oleva numero vaikka negatiivinen tai nolla, ja siinä sitä ollaankin.

Kuten nokkelimmat jo ehkä huomasivat, käytännössä pelaajalla on käytössään yksinkertainen konekieli. Lattialla olevat laatikot vastaavat muistirekisterejä, ja duunarille suoritettavaksi purkitettavista käskyistä muodostuu yksinkertainen ohjelma, jolla numerot prosessoidaan annetun algoritmin mukaisesti.

Human Resource Machinen oppimiskäyrä on saatu aika hyvin kohdalleen. Alussa vielä mennään simppelisti, kun tavoitteena on vaikka lähettää peräkkäisten lukujen summa eteenpäin, mutta pian haaste nousee aika tavalla. Kun tehtävänä on vaikka siirtää outboxiin kaikki Fibonaccin lukujonon alkiot (numerosarja, jossa aina kaksi peräkkäistä lukua summataan yhteen muodostamaan seuraava), jotka ovat pienempiä kuin saatu syöte, alkaa tulla hiki. Varsinkin kun syötteenä tulevat numerot ovat satunnaisia, joten ratkaisujaan ei voi koodata tiettyjä tapauksia varten, vaan niistä täytyy suunnitella yleisiä.

Human Resource Machine on mainio peli. Se näyttää veikeältä, on helposti lähestyttävä ja todellakin opettaa kuin huomaamatta huumorin avulla algoritmisen ajattelun perusteita ja käsitteitä. Eihän tämän jälkeen toki vielä mitään assembly-koodia osaa lähteä paukuttamaan, mutta ei se ole edes tavoitteena.

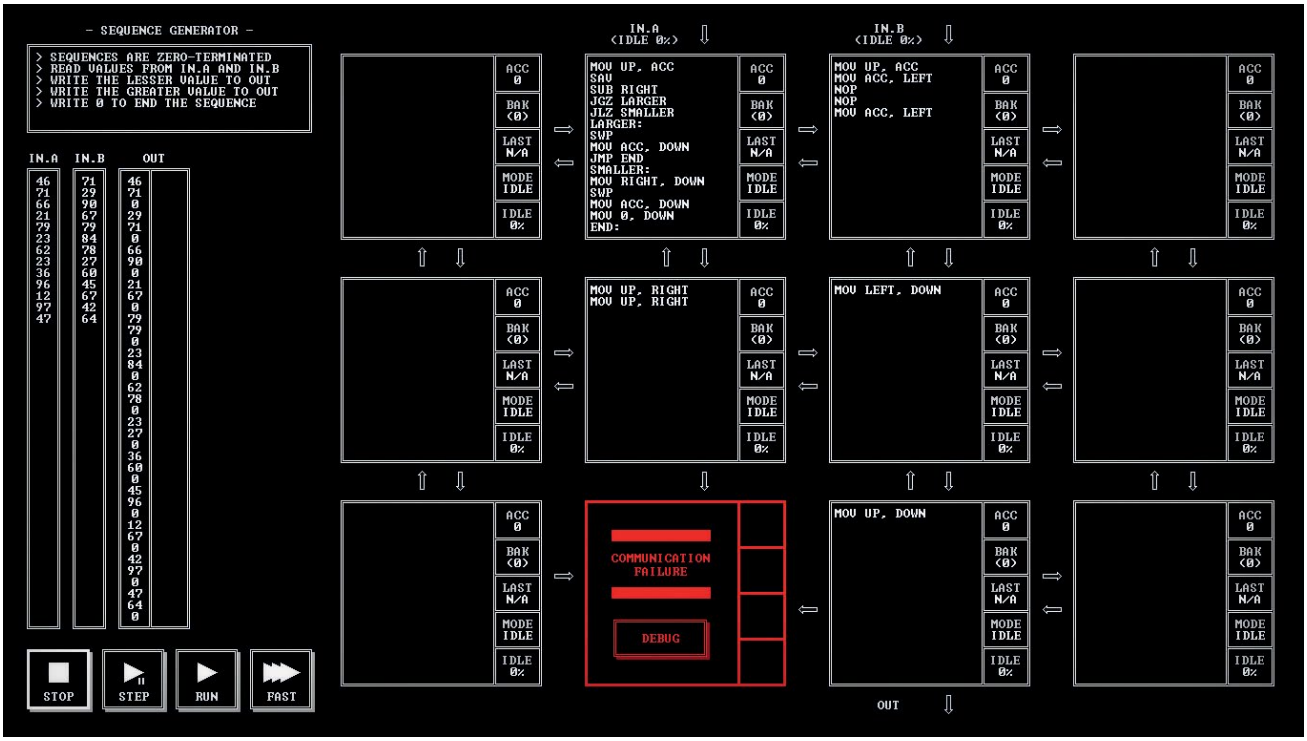
TIS-100

Jos Human Resource Machine pyrki tavallaan huijaamaan pelaajan opettelemaan konekieltä, TIS-100 heittää haasteen suoraan naamalle. Pelissä pelaaja on saanut käsiinsä unohdetun 80-luvun kotimikron, TIS-100:n. Vuosien varrella kone on kärsinyt sen verran, että kaikki muut sen ohjelmat ja mahdollisuudet ovat tuhoutuneet, paitsi koneen yksinkertainen konekielinen käskykanta.

Pelin mukana tulee PDF-manuaali, jossa ihan oikean ohjelmointioppaan tavoin selitetään käskyt sekä kehoitus tulostaa manuaali. Sitä kun tulee tarvitsemaan. Edessä on useampi kymmenen koodaushaastetta, jossa yksinkertaisen käskykannan avulla tulee rakentaa uudelleen monimutkaisia korkeamman tason toimintoja.

Peli tekee pieniä kädenojennuksia pelattavuuden suuntaan, sillä esimerkiksi tietokoneen muistirekisterit on ladottu kaksiulotteiseksi taulukoksi, jossa dataa siirrellään fyysisten ilmansuuntien mukaan. Tyypillisessä haasteessa saadaan yhdestä tai useammasta syötteestä lukuja, jotka sitten pitää käsitellä ja lähettää eteenpäin annettujen sääntöjen mukaisesti.

Hommaa mutkistaa se, että kaikkia rekisterejä ohjelmoidaan paikallisesti ja suoraan. Muihin rekistereihin ei voi



TIS-100 tarjoaa mahdollisuuden koodata hyvinkin uskottavaa ja wanhaa assembly-konekieltä, vieläpä viihdyttävästi – vaikka sitä ei ehkä kuvasta usko.

viitata, vaan ainoastaan luvut liikkuvat niiden välillä. TIS-100 ei myöskään osaa itse synkronoida ohjelmia mitenkään, vaan eri tahtiin kulkevat aliohjelmat täytyy synkronoida itse käsin heittelemällä väliin nollakomentoja. Niin – ja kuhunkin rekisteriin mahtuu 15 rivin edestä koodia. Ei ole helppoa, ei.

Siinä missä Human Resource Machinen vaikeuskäyrä on mukavan loiva ja pelikokemus sujuva, TIS-100 ottaa luulot pois jo aika varhaisessa vaiheessa. Jos konekielellä koodaaminen ei suju, pelistä ei saa hirveästi irti. Kyse ei toki lopulta ole suunnattomasta teknisestä haasteesta, sillä käskykanta on pieni, rajoittunut ja suoraviivainen. Niinpä oikea haaste tuleekin siitä, että asioita täytyy tehdä yhtä aikaa yksinkertaisesti ja monimutkaisesti. Erilaisten elementaalioperaatioiden synkkaukset ja ohjelman jakaminen ovat ongelmia, joiden kanssa joko oppii elämään tai lopettaa pelaamisen.

Tätä kautta pelissä pääsee myös hyvin kokemaan käytännössä, millaista koodaaminen oli vanhoina hyvinä aikoina. Kun sain veteraanipelaajalta vinkin pudottaa koodistani pois pilkut ja käyttää yksikirjaimisia hyppykoodeja kuvaavien sanojen sijaan, jotta riveille – ja täten rekisteriin – mahtuisi enemmän koodia, olin kuin ällikällä päähän lyöty.

TIS-100 onkin hyvin palkitseva ja addiktoiva pelikokemus. Jokainen onnistunut suoritus tuntuu voitolta, ja suorituksen jälkeen peli näyttää kuinka omat kaverit ovat saman ongelman selvittäneet. Jos oma koodi ratkoi ongelman kahdeksalla rutiinilla ja 1359 syklillä ja kaveri selvisi siitä kuudella rutiinilla ja 1200 syklillä, niin kyllähän se potkii optimoimaan ja tuunaamaan omaa koodia. Kyllä minä sille vielä näytän!

CodeCombat

CodeCombat on hyvä esimerkki viime aikoina suosiota ja näkyvyyttä keränneistä peleistä, joiden tarkoitus on opettaa käytännön ohjelmointia pelaamisen lomassa. Pelaaja ohjastaa roolipelin sankaria, mutta ei kuitenkaan peliohjaimella tai näppäimistöllä vaan ohjelmoimalla.

Idea on mitä mainioin ja tarjoaa vähän syvällisemmän algoritmisen ajattelun sovelluksen kuin TIS-100 tai Human Resource Machine. Niissä ongelmat ilmoitetaan valmiina algoritmeina ja pelaajan vastuulla on toteuttaa ne annetun käskykannan avulla. Nyt pitää jo itse vähän kyetä miettimään ja purkamaan sumeita ongelmia (”kulje labyrintin läpi kolikoita keräten ja hirviöitä tappaen”) loogisiksi askelmiksi, joita voidaan sitten seurata. CodeCombat pätee myös sillä, että toisin kuin aiemmissa peleissä, CodeCombatia pelataan viidellä oikean maailman ohjelmointikielellä, joita pelaaja sitten oppii pelatessaan käyttämään.

Kun koodi saa miekan huitomaan ja örkit kuolemaan, koodauksen opettelu tuntuu oikein viihdyttävältä puuhalta!

Läpäistessään tasoja pelaaja kerää kokemuspisteitä, pelin sisäistä valuuttaa ja hahmolleen myös uusia varusteita. Tämäkin puoli pelistä sopii hyvin ohjelmoinnin maailmaan, sillä mainiona ideana erilaiset varusteet laajentavat pelaajan käytössä olevaa käskykantaa hyvin samalla tavalla kuin jos koodatessaan toisi ohjelmaansa mukaan ulkopuolisia luokkia. Esimerkiksi miekan käteensä ottamalla hahmo oppii self.attackNearestEnemy()-tyylisiä käskyjä, kun taas vasaralla voi rakennella ansoja ja esteitä.

Vaikeuskäyrä on toteutettu mainiosti. Peli etenee aivan perusteista alkaen askel kerrallaan läpi aliohjelmien, silmukoiden, tiedon välittämisen oliosta toiseen ja niin edelleen. Kun huomioidaan, että CodeCombat aloittaa aivan perusteista, se on yllättävän syvä paketti. Tarjolla on toista sataa erilaista kenttää, joten ohjelmoinnin eri osa-alueet ja piirteet tulevat taatusti tutuiksi. Loppua kohden mennään jo aika korkean tason juttuihin.

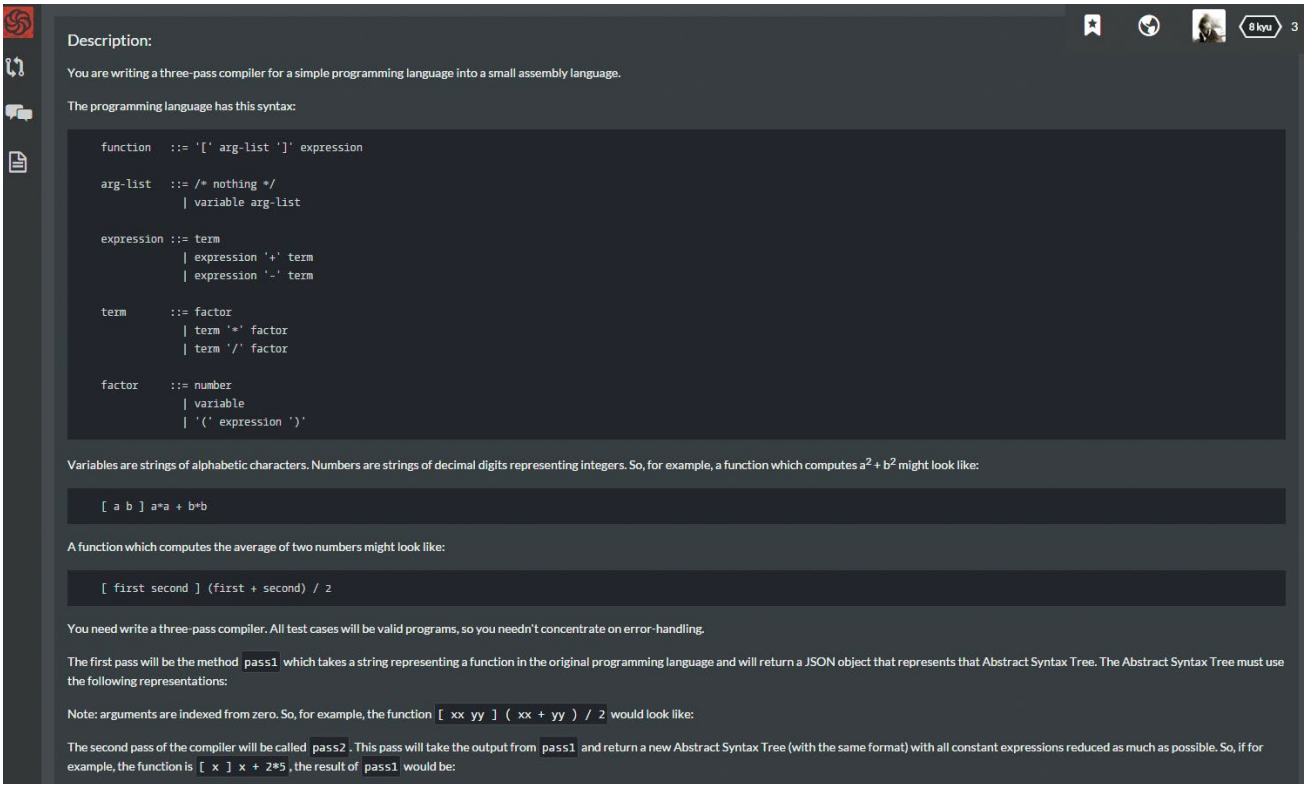
On toki huomioitava, että CodeCombat pyörii skriptauskielten ympärillä ja tarjoaa täten verrattain rajoittuneen kokonaiskuvan ohjelmoinnin maailmaan. Suurempien kokonaisuuksien kasaaminen ja ohjelmistosuunnittelu eivät tietenkään kuulu sen mittakaavaan, mutta se hoitaa lestinsä mainiosti. Se tekee algoritmisen ajattelun opettelemisesta ja harjoittelemisesta helppoa ja sujuvaa puuhaa. Kun on opetellut perusteet CodeCombatilla, voi todennäköisesti siirtyä verrattain pienellä vaivalla muiden kielten pariin tekemään suurempia juttuja, mikäli mielenkiintoa riittää.

CodeWars

Niistä suuremmista jutuista ja muista kielistä puhuttaessa pitää vielä mainita CodeWars, joka eroaa niistä aiemmista peleistä vaikka sillä, että se ei oikeastaan ole peli. Se ei ole peli, mutta se käyttää pelillisiä elementtejä.

Pelillistäminen on ilmiö, jossa pyritään tekemään tylsistä tai työmäisistä asioista hauskempia lisäämällä mukaan elementtejä peleistä. CodeWars on koodaustreenaaja, jossa itsepuolustuslajien hengessä edustetaan klaania tai jotain muuta tahoa ja kerätään sille kunniaa suorittamalla erilaisia tehtäviä.

Pelin kielivalikoima on testissä olevien pelien laajin, sillä tarjolla on jo nyt pino oikeita ohjelmointikieliä, ja tulossa on koko ajan lisää. Toisin kuin testin muita pelejä, CodeWarsia ei ole suunniteltu edes täysille aloittelijoille. Kun aluksi on valinnut opetteluun haluamansa kielen, täytyy seuraavaksi todistaa hallitsevansa sen perusteet korjaamalla rikkinäisiä ohjelmia toimiviksi. Mistään suunnattomista haasteista ei ole kyse, sillä omassa C#-testissäni esimerkiksi piti huomata, että eräs aliohjelma yritti palauttaa muuttujaa, joka oli määrittelyssä kirjoitettu



CodeWarsin tehtävät alkavat perustason jutuista, mutta tasojen noustessa myös haasteet kasvavat aika reippaasti. Loppupäässä saa jo koodata tosissaan.

pienellä alkukirjaimella kun taas palautuksessa suurella.

Kun rima on ylitetty, edessä on valtava koodaushaasteiden allas, johon sitten sukeltaa. CodeWars pyörii pitkälti crowdsourcing-pohjalta, eli yhteisö saa itse kehittää ongelmia ja arvioida niiden vaikeustasoa ja soveltuvuutta. Itsepuolustuslajien eri tasojen mukaan jaotellut tehtävä ovat jo aluksi sangen kinkkisiä ja todella monipuolisia. Haasteet on jaettu laajoihin kategorioihin, joista voi valita juuri itseään kiinnostavia aihepiirejä ja sitten lähteä leikkimään sankarikoodaajaa.

Yhteisö on mukana myös keskustelemassa ongelmista, joten jos ajautuu pulman kanssa umpikujaan, voi kysyä muilta neuvoja tai vaihtoehtoisesti tutoroida muita saitin käyttäjiä.

Vaikka CodeWars ei olekaan yhtä viihdyttävä tai välittömän ihastuttava kuin testin muut pelit, omasta perspektiivistäni se on ehkä antoisin. Olen itse maisterivaiheen tietojenkäsittelytieteiden opiskelija, ja omat opintoni keskittyvät nykyään pääasiassa ohjelmistotuotantoon ja muihin teoreettisempiin juttuihin, joten on oikein mukava päästä venyttelemään koodauslihaksiaan ja tekemään jatkuvasti uusiutuvia ja sopivan haastavia pähkinöitä.

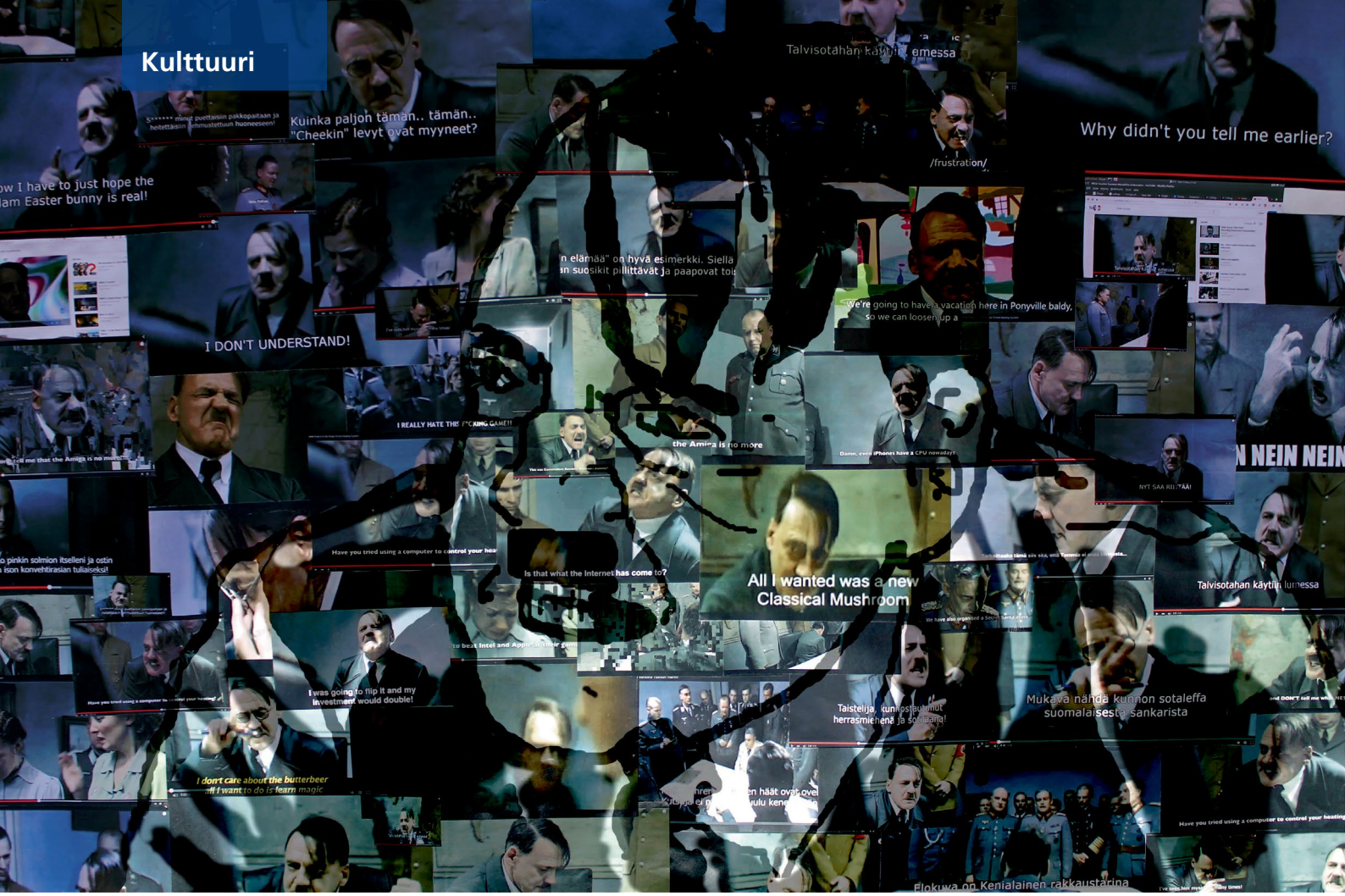
CodeWars sopii myös hyvin nyt tarjoilemamme loogisen jatkumon päätepisteeksi. Human Resource Machine on hyvä koukutin algoritmisen ajattelun maailmaan ja ohjelmointiin mukavan visuaalisesta perspektiivistä esitettynä. CodeCombat opettaa sitten oikean ohjelmoinnin perusteet. Kun ne ovat hallussa, taitojaan voi hioa ja uusia kieliä opetella kätevästi vaikka CodeWarsin parissa.

Human Resource Machine
Tuetut kielet: oma alkeellinen konekieli
Alustat: Mac, Windows, Wii U
Hinta: 9,99 €
Lisätietoja: tomorrowcorporation.com/humanresourcemachine

TIS-100
Tuetut kielet: oma assembler-konekieli
Alustat: Linux, Mac, Windows
Hinta: 6,99 €
Lisätietoja: www.zachtronics.com/tis-100/

CodeCombat
Tuetut kielet: Python, JavaScript, CoffeeScript, Clojure ja Lua
Alustat: selainsovellus
Hinta: Ilmainen, mutta tarjolla on myös kuukausimaksullinen versio, joka sisältää enemmän erilaisia hahmoja ja muita pieniä bonuksia.
Lisätietoja: codecombat.com

CodeWars
Tuetut kielet: CoffeeScript, JavaScript, Python, Ruby, Java, Clojure, Haskell, C# (tulossa F#, Obj-C, php, C, C++)
Alustat: selainsovellus
Hinta: ilmainen
Lisätietoja: codewars.com



”Pistä natsi asialle ja mene itse perässä!” **Hitler-videomeemien anatomia**

Sosiaalisessa mediassa leviävät Hitler-videomeemit perustuvat Oliver Hirschbiegelin ohjaamaan, Adolf Hitlerin viimeisiä päiviä esittävään elokuvaan Perikato (Der Untergang, 2004). Videomeemejä alkoi syntyä vuoden 2005 jälkeen, ja ne ovat kymmenen vuoden ajan olleet erityisen suosittuja YouTube-videovälityspalvelussa.

Teksti: Petri Saarikoski

Kranaatit räjähtelevät Berliinin kaduilla, rakennuksia sortuu ja sirpaleita lentelee kaikkialla. Neuvostankit rynnäköivät eteenpäin etsien läpimurtokohtaa yhä ohuemmiksi käyvästä puolustuslinjasta. Kaaoksen keskellä keskustan hallintokorttelissa vartiosotilaat vetäytyvät suojaan komentobunkkeriin ja pamauttavat oven kiinni perässään. Syvällä maan sisällä alkaa tilannekatsaus Adolf Hitlerin johtohuoneessa.

Natsi-Saksan diktaattori on erittäin vihainen. Ulkona riehuva sota saa unohtua, sillä käsillä on huomattavasti vakavampi tilanne. Suomen vuoden 2015 edustakuntavaalien lopputulos on johtanut katastrofiin ja keskustan ”juntit” ovat nousemassa valtaan. Punamullan sijasta Suomeen olisi luvassa ”juntti-porvarihallitus”. ”Demokratia on puhdasta pelleilyä!”, Hitler raivoaa. ”Stalin sentään tajusi, että kansalta ei pidä kysyä yhtään mitään!” Raivokohtaus on tallennettu YouTubeen illalla 19. huhtikuuta 2015 eli heti vaalituloksen selvittyä.

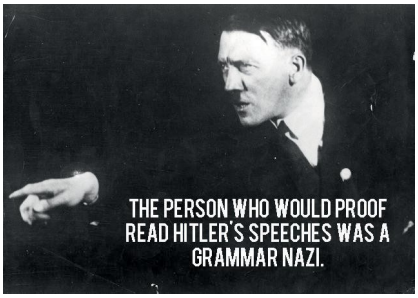
Hitler-videot kotimaisen median hampaissa

Tapaus on hyvä esimerkki ”Hitler kuulee” -nettivideoiden tavasta ottaa parodian keinoin kantaa ajankohtaisiin tai muuten vain mielenkiintoisiin ilmiöihin. Hitler-nettivideot ovat tyypillisiä sosiaalisen median meemejä eli nopeasti verkossa leviäviä, tietylle idealle perustuvia kulttuurimuotoja.

Videot perustuvat pääsääntöisesti Perikato-elokuvan kohtaukseen, jossa Adolf Hitler saa hillittömän raivokohtauksen ymmärrettyään, että sota on hävitty. Kohtaus tekstitetään uudelleen ottamaan kantaa ajankohtaisiin, yleisöä puhuttaviin tapahtumiin. Hitler-videoiden nettivirta jatkuu vuosien taakse. Tällä hetkellä YouTubessa saatavilla olevia erikielisiä versioita löytyy pikahaun perusteella sadoittain. Suosituimmilla videoilla on miljoonaluokan yleisö.

Suomessa valtamediat kiinnostuivat ilmiöstä keväällä 2015, kun toisen maailmansodan päättymisestä tuli kuluneeksi 70 vuotta. Hitlerin viimeiset päivät Berliinissä olivat sodan loppuvaiheen tunnetuimpia tapahtumia ja Yleisradio esitti tähän liittyen Perikato-elokuvan uusintana 24. huhtikuuta. ”Vieläkö [Perikato-elokuvaa] ja sen keskeistä kohtausta voi katsoa vakavana, huvittumatta?” pohti Helsingin Sanomat. On kiinnostavaa, että valtamediat eivät olleet laajemmin kir-

Charlie Chaplin parodioi Hitleriä Diktaattori-elokuvassa. Mielenkiintoisella tavalla Diktaattorista otettiin vaikutteita Iron Sky -elokuvaan, josta löytyy niin ikään oma Hitler-videoita parodioiva kohtaus – monella tavalla parodia on kiertänyt historiallisen ympyrän. Lähde: Wikipedia.



Kaksi esimerkkiä Hitler-meemeistä: tekstitetty kuva ja gif-animaatio. Molempien tekijät tuntemattomia.

joittaneet Hitler-videoista kun elokuva oli esitetty aikaisemmin televisiossa (2004, 2008, 2012).

Laajaa mediahuomiota aiheelle syntyi, kun Rakkautta & Anarkiaa -elokuvafestivaali käytti meemiä tapahtumansa markkinoinnissa vuonna 2014. Meemeille tyypilliseen tapaan kommentointi oli kehittynyt metatasolle, sillä festivaalin trailerissa esiintyi Adolf Hitlerin sijaan ruotsia puhuva Mannerheim, joka raivostui oman suurelokuvansa kariutumisesta. Suositttu video lähti sosiaalisessa mediassa nopeasti viralisoitumaan ja lopulta traileri sai huomattavasti enemmän huomiota kuin tapahtuma itse. Luonnollisesti trailerin rinnalle alkoi heti ilmestyä videoita, joissa parodioitiin traileria. Pian ilmestyi videoita, joissa parodioitiin trailereita, jotka parodioivat alkuperäistä traileria. Muutaman viikon päästä kommentointi kyllääntyi ja kiinnostus laimeni.

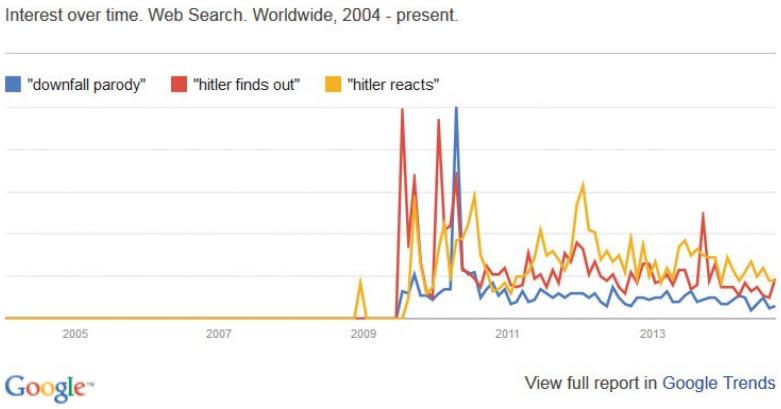
Olen itse tutkinut Hitler-videomeemejä osana sosiaalisen median historiaa. Miten Hitler-videot ovat syntyneet, ja mitä ne kertovat sosiaalisen median muodonmuutoksesta? Ymmärtääksemme niiden merkitystä ja suosiota meidän on pakko kääntää katseemme menneisyyteen. Miten ja millä tavoin Hitleristä tuli Internet-parodioinnin kohde ja miksi hänelle on helppoa nauraa?

Hitler-karikatyyrin synty

Adolf Hitleristä (1889–1945) ja hänen ajastaan on kirjoitettu hyllymetreittäin tutkimuksia, yleisteoksia, selvityksiä ja dokumentteja. Lehtiartikkelien määrää tuskin kukaan pystyy laskemaan. Siitä lähtien, kun Hitler ensimmäisen kerran ilmestyi julkisuuden parrasvaloihin 1920-luvulla, hän tuli tunnetuksi erityisesti taitavana ja manipuloivana puhujana. Poliitikkona Hitler oli synnynnäinen ja laskelmoiva näyttelijä, joka pystyi omaksumaan tarpeen mukaan lukuisia eri rooleja.

Hitleriä käsittelevien parodioiden ja satiirien juuria voidaan jäljittää aina hänen poliittisen uransa alkuvaiheisiin 1920- ja 1930-luvulla. Mahtipontiset, usein samalla kaavalla vedetyt julkiset esiintymiset sekä peittelemättömän aggressiivinen ja kansallismielinen propaganda olivat omiaan synnyttämään poliittisia pilapiirroksia ja humoristisia kannanottoja. Saksan vasemmistopiireissä julkaistiin myös Hitleriä satiirisesti käsitteleviä propagandaelokuvia vielä 1930-luvun alussa. Toisen maailmansodan sytyttyä toiminta muuttui vakavampaan suuntaan ja yhdistyi Hitlerin vastustajien poliittiseen propagandaan. Nimikkeistä tärkein ja tunnetuin oli Charlie Chaplinin Yhdysvalloissa ohjaama Diktaattori (The Great Dictator, 1940), jossa esimerkiksi Hitlerin mahtipontinen esiintymistyyli joutui pilailun kohteeksi. Todellinen humoristisen ja satiiristen esitysten vyöry lähti liikkeelle heti Yhdysvaltojen liityttyä Saksan-vastaiseen sotaan liittoutuneiden rivissä.

Toisen maailmansodan päättyminen Saksan täydelliseen tappioon johti lopulta siihen, että Hitlerin karikatyyrinen hahmo jäi elämään oma elämäänsä populaarikulttuurissa. Hahmona Natsi-Saksan diktaattori kiinnosti suurta yleisöä historiallisena henkilönä, mutta hän kelpasi myös lukemattomien vitsien, sketsien ja komediapätkien aiheeksi. Hitler eli niin sarjakuvissa, televisiosarjoissa kuin elokuvissa. Hitler-hahmo on kytketty komedian keinoin ottamaan kantaa ajankohtaisiin yhteiskunnallisiin tai kulttuurisiin ilmiöihin. Esimerkkinä mainittakoon komediaryhmä Monty Pythonin Britannian äärioikeistoa pilkkaava Hitler in England -sketsi vuodelta 1970. Toisena esimerkkinä South Park -animaatiosarjan (1997–) Hitleriksi pukeutunut rääväsuinen



Google Trendsin tilasto, jossa hakutulosten valossa näkyy miten Hitler-videoiden maailmanlaajuinen suosio on kehittynyt.

Cartman, joka vaihtelevalla menestyksellä näytti keskisormea ihmisten kaksinaismoralismille ja tekopyhyydelle.

Verkkokulttuurin ja internetin yleistyttyä 1980- ja 1990-luvulta lähtien Hitleristä tuli harrastajien verkkovitsien peruskauraa. Tyypillisimpiä esitystapoja oli käyttää alkuperäistä, historiallista kuvamateriaalia ja tekstittää sitä uudelleen tarpeen mukaan tai vaikkapa pistää Aatu pomppimaan laaman selässä gif-animaatiossa. Vain käyttäjien luova hulluus oli rajana sille, miten hahmoa voitiin käyttää humoristisen nettikulttuurin välineenä.

Nettiparodia astuu näyttämölle

Perikato-elokuva asettuu tässä populaarikulttuurissa mielenkiintoiseen saumakohtaan. Eurooppalaisena suurproduktiona toteutettu Perikato (Der Untergang, 2004) on ylipäätään yksi harvoja saksalaisia toista maailmansotaa käsitteleviä ison budjetin elokuvia – muita esimerkkejä ovat Das Boot vuodelta 1981 ja Stalingrad vuodelta 1992. Tässä vaiheessa sodasta oli kulunut jo niin kauan, että kiinnostus aihetta kohtaan oli noussa yhdistyneessä Saksassa. Hitlerin henkilöhahmon nostaminen elokuvan pääaiheeksi synnytti vilkasta ja kriittistä keskustelua. Katsojia ja kriitikoita arvelutti erityisesti elokuvan tapa inhimillistää absoluuttisen pahan symbolina tunnettua diktaattoria. Vihaa ja voimaa puhkuvan karismaattisen Hitlerin tilalla nähtiin henkisen ja fyysisen romahduksen partaalla vavissut ukko, joka herätti lähinnä säälin tunteita.

Perikato oli kaupallinen menestys ja varsinkin pääosan esittäjä Bruno Ganzin roolisuoritusta ylistettiin laajasti. DVD-julkaisun ilmestyttyä suosittu elokuva lähti nopeasti leviämään verkossa piraattiversiona. Näppärät nettiharrastajat ottivat elokuvan luovan leikittelyn välineeksi. Erityisen paljon kiinnostusta herätti elokuvan kuuluisa raivokohtaus. Mitäpä jos tämä kohtaus tekstitettäisiin uudelleen? Huippusuosituksi noussut YouTube-videolatauspalvelu aloitti toimintansa vuonna 2005 ja tarjosi parhaan mahdollisen alustan parodiavideoiden levitykselle.

On mahdotonta sanoa mikä oli ensimmäinen Perikato-elokuvaan perustuva Hitler-videomeemi, koska varhaisista viirityksistä on jäänyt vain sattumanvaraisia jälkiä. Nettimeemejä arkistoivien palvelujen (esimerkiksi knowyourmeme.com) mukaan ainakin yksi varhaisimmista tunnistetuista esimerkeistä oli nimimerkki ”DReaperF4”:n YouTubeen 10.8.2006 lataama Sim Heil: Der untersim (espanjankieliset tekstitykset), joka koski Microsoftin Flight Simulator X -pelin demoversion huonoja ominaisuuksia. Video nousi varsin suosituksi ja sai rinnalleen englanninkielisiä versioita. Ilmiön varhaisvaiheen videot, joita katsottiin miljoonia kertoja, ottivat erityisesti urheilu- ja pelikulttuurien ilmiöt hampaisiinsa: ”Hitler gets banned from Xbox Live” (2007), ”Ronaldo leaves Machester United for Real Madrid” (2008), ”Usain Bolt Breaks 100m World Record” (2009). Kotimaisista varhaisvaiheen videoista suosituimpia oli YouTubeen 29.9.2009 ladattu ”Hitlerin olut-ongelma”. Natsikenraalien myöhästyneen kaljanhakureissun jälkipuintia oli katsottu syyskuuhun 2015 mennessä yli 900 000 kertaa.

Monien Hitler-videoiden alkuperäisten tekijöiden jäljittäminen on hyvin vaikeaa, koska eri versioita alettiin kopioida ja kääntää surutta eri kieliversioille. Joka tapauksessa vuoteen 2009 mennessä videot olivat nousseet kansainvälisesti niin suosituiksi, että niille alettiin perustaa omia nettiyhteisöjä ja -ryhmiä. Seuraavaan vuoteen mennessä oli syntynyt näihin videoiden erikoistunut ”Untergangers”-alakulttuuri, jonka piirissä videomeemejä pyrittiin kehittämään aikaisempaa kunnianhimoisempaan suuntaan. Pelkkien uudelleentekstitysten sijaan videot voitiin luovasti rakentaa täysin uusiksi ja sisällyttää niihin klippejä elokuvan muista kohtauksista tai laittaa niitä vuoropuheluun muun mediakulttuurin kanssa. Esimerkkinä mainittakoon Hitler-versiot korealaisen rap-artisti PSYn ”Gangnam Style” -videosta (2013). Undergangers-harrastajista ehkä tunnetuin on Isosta-Britanniasta kotoisin oleva Stacy Lee Blackmon, joka tunnetaan myös nimimerkillä ”Hitler Rants Parodies” tai ”Elite Prime”. Hän on julkaissut yli 1100 Hitler-videomeemiä.

Videomeemit tuotantoyhtiön hampaissa

Videot joutuivat melko nopeasti myös sensuurin kohteeksi. Perikato-elokuvan ohjaaja Oliver Hirschbiegel oli haastattelujen mukaan tietoinen videoista jo melko varhaisessa vaiheessa ja piti niitä lähtökohtaisesti myönteisenä osana Internetin remix-kulttuuria. Haastattelun perusteella Hirschbiegel suhtautui alusta asti meemeihin hyväksyvästi, koska ne sopivat hänen mukaansa elokuvan alkuperäiseen tarkoitukseen: Adolf Hitlerin sisäpiirin toiminnan mielettömyyden kuvaamiseen. Sitä paitsi monet videoista olivat hänen mukaansa äärimmäisen ”hauskoja”. Haastattelussa hän vihjaisee, että videot lisäsivät selvästi elokuvan tunnettavuutta ja olivat näin ollen myös taloudellisesti hyödyllisiä. Tuotantoyhtiö sen sijaan oli täysin toista mieltä. Vastatoimet kulminoituivat vuonna 2010, jolloin YouTube joutui liukuhihnalla poistamaan videoita palvelustaan. Monet varhaisvaiheen videoista hävisivät tässä vaiheessa lopullisesti, mutta suosituimmat klipit yksinkertaisesti vain ladattiin palveluun uudelleen. Tuotantoyhtiön perustelujen mukaan poistot olivat oikeutettuja, koska

se piti niitä mauttomina ja epäasiallisina. Todelliset syyt lienevät olleet tekijänoikeudellisia.

Sensuuriyritykset kääntyivät lopulta tuotantoyhtiötä vastaan. Vuoden 2010 aikana Hitler-videoiden maailmanlaajuinen suosio nousi kattoon, ja rajoitustoimet näyttivät päin vastoin vain lisäävän tietoisuutta niiden olemassaolosta. Loppuvuodesta 2010 tuotantoyhtiö nosti kädet pystyyn ja sensuuripolitiikkaa höllennettiin. Yhtiö alkoi jopa mainostaa elokuvaa videoiden yhteydessä.

Veteraaninäyttelijä Bruno Ganz ei ole juurikaan kommentoinut videomeemejä julkisuudessa. Tosin YouTubessa julkaistiin kolmisen vuotta sitten lyhyt videohaastattelu, jossa Ganz sanoi ihailevansa harrastajien ”luovuutta”. Aluksi hänen oli ollut hyvin vaikea hyväksyä parodioita, koska oli panostanut niin valtavasti roolisuoritukseensa. Hänen mukaansa parodioissa ja huumorissa ei ollut sinällään mitään vikaa ja Ganz toki ymmärsi, että meemivideoiden ansiosta hän oli nyt näyttelijänä ”tunnetumpi kuin koskaan”. Uskottavalta kuulostavat kommentit paljastuivat lähemmin tarkasteltuna huvittavaksi metakommentoinniksi: haastattelun tekstitykset eivät vastanneet saksaksi puhuvan näyttelijän käsittelemiä asioita. Video osoittautui parodiaksi parodiasta.

Kriittisiä ajatuksia meemeistä

Hitler-meemivideoihin liittyvää kritiikkiä ei voi kuitenkaan täysin sivuttaa. Miten Hitler soveltuu parodioinnin lähteeksi? Kysymystä kannattaa hienosäätää ja tarkentaa: voinko parodioida ajankohtaista ilmiötä ja valita vitsailuni työkaluksi Hitlerin? Nettikeskustelukulttuurille tyypilliseen yleistykseen viitaten voisi kysyä: ”Onko pakko heittää Hitler-kortti pöytään, kun haluan nauraa?”

Aiemmin mainittu Helsingin Sanomien kirjoitus, jossa pohdittiin voiko Perikato-elokuvan tunnetuimpaan kohtaukseen suhtautua enää ”vakavasti” on mielenkiintoinen. Kysymys viittaa siihen, että todellisiin historiallisiin tapahtumiin perustuva elokuva olisi jotenkin menettänyt merkitystään ja sosiaalisen median ruokkimat videomeemit olisivat nousseet etualalle.

Parodian hauskuus syntyy leikittelystä, jossa katsoja pystyy tunnistamaan

"Hitler kuulee Guggenheimista" (2012)

"Hitler kuulee Talvivaarasta" (2013)

"Hitlerin olut-ongelma" (2009)

käsiteltävän, ajankohtaisen ilmiön uudessa ympäristössä. Hitler-videomeemien taustahistoria paljastaa, että diktaattorille naureskelulla on pitkät perinteet. Leikittely toimii näin syntyvästä kontrastista, jossa alkuperäistä tapahtumaa voi käsitellä omituisella mutta tunnistettavalla tavalla. Leikittely kuitenkin väistämättä pinnallistaa kohdettaan. Hitler oli 1900-luvun tuhoisimman sodan pääarkkitehti, joka aiheutti Euroopassa enemmän tuhoa kuin oli koskaan koettu.

Mitä me oikeasti ajattelemme kun katsomme Hitler-videoita? Voiko pahuudelle nauraa? Vai naurammeko me itsellemme? Tulkinnat voivat olla varsin ristiriitaisia, mutta samalla moniulotteisia.

ARMOTTOMAT PIKSELIT: LÄNNENPELIEN JÄLJILLÄ

Villin länsi teemalla on tuotettu massiivinen määrä populaarikulttuuria niin elokuvista iskelmiin kuin sarjakuviinkin, eivätkä pelit ole mikään poikkeus. Lautapeleistä, huvipuistojen mekaanisista kaksintaisteluista ja flippereistä oli luonteva siirtyä bittimaailmaan, kun digitaaliset pelit alkoivat yleistyä kodeissa ja pelihalleissa 1970-luvun kuluessa.

Teksti: Tero Heikkinen ja Markku Reunanen

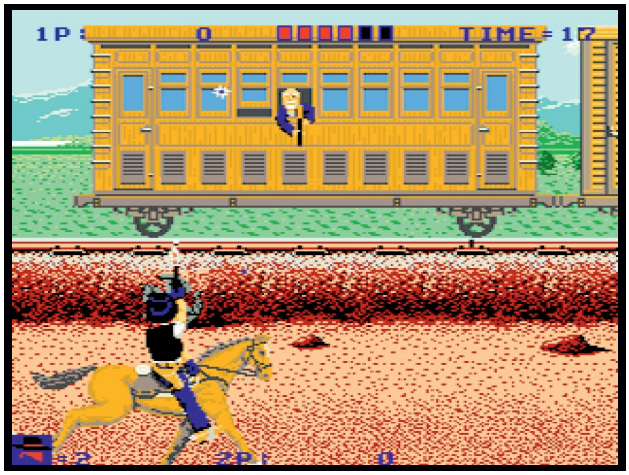
Kuvat: Manu Pärssinen, Tero Heikkinen

Villi länsi oli vahvasti tuotteistettu fantasiamaailma jo 1900-luvulle tultaessa, sillä romantisoitua kirjallisuutta sekä showesityksiä oli tehty pitkään. Etenkin elokuvien kautta lyötiin lukkoon tunnetuimmat lännenkliseet: yksinäinen sankari puhdistaa pahiksen hallitseman, vellihousujen kansoittaman kaupungin, nitistää rikolliset ja poistuu voittajana paikalta. Vaarallisin vastustaja kukistetaan tarinan huipennuksessa, lopun tiivistunnelmaisessa kaksintaistelussa. Lännenteemoja muokattiin, rikastettiin ja kyseenalaistettiin elokuvissa yli puolen vuosisadan ajan ennen ensimmäisiä kotitietokoneita ja pelikonsoleita, joten tilaisuuden tullen valmista materiaalia oli tarjolla runsaasti.

Flipperit ja pelihallit

Viihteellistetty länsi on jo itsessään varsin pelimäinen: korttipelien ja ruletin ohella onnenpyörää ja yksikätistä rosvoa vilahtelee elokuvissakin. Kukapa ei olisi nähnyt lännenelokuvaa, jossa virnuilevan pahiksen pokerikäsi paljastuu sittenkin sankaria huonommaksi. Rajaseudulla tarvittu mytologinen taituruus heijastui suoraan erilaisiin mekaanisiin peleihin – alkeellisia flippereitä sekä rahapelejä koristeltiin intiaani- ja rodeoteemoilla jo 1900-luvun alussa. Revolverisankarien lisäksi toistuvia teemoja etenkin flippereissä olivat puolipukeiset intiaanineidot ja lehmitytöt.

Ensimmäiset varsinaiset pelihallit koostuivat huvipuistolaitteista ja mekaanisista peleistä. Elektroninen arcade-kulttuuri oli näiden mekaanisten laitteiden suoraa jatketta ja toisintoa: esimerkiksi lännen kaksintaistelut sekä reaktiopelit olivat tuttua tavaraa jo ennen mikropiiriaikaa. Taistelupari saattoi vertailla vetämisnopeuttaan reaktiokoneen äärellä tai yksittäinen pelaaja koetella kuntoisuuttaan luonnollisen kokoista sähkömekaanista mallinukkea vastaan. 1960-luvun peleissä *Mr. Quick Draw* tai *Mr. Top Gun* aidolta näyttävä lännenmies kohotti aseensa pelaajaa kohti. Eipä ihme, että vuoden 1973 Westworld-elokuvassa fantasioitiin robottihahmojen ennen pitkää kapinoivan ihmisiä vastaan.



Laukkaa ratsu reima: Outlaws ja Express Raider.

Digitaaliviihde laukkaa koteihin

1970-luvun ensimmäisten konsolipelien juuret ovat vastaavasti aikakauden kolikkopeleissä: *Pong*-klooneja markkinoitiin eri urheilulajeina tenniksestä jalkapalloon, ja pienillä muutoksilla sama mekaniikka sovitettiin myös eräänlaisiksi lännenpeleiksi. Etenkin Taiton *Gunfightia* kloonattiin kotikonsoleilla ja -tietokoneilla ahkerasti vielä pitkään 1980-luvun jälkipuolelle asti. Ruudun laidoilta toisiaan ampuvat tikku-ukot sopivatkin hyvin varhaisten laitteiden kyvyille, eikä kahden pelaajan mittelöön tarvinnut rakentaa edes tekoälyä. Gunfightin Pong-historiasta kertoo osaltaan se, että nelikulmaiset luodit kimpoavat kentän ylä- ja alareunasta.

Pelihalleissa ja 2600-konsolillaan menestynyt Atari ei jättänyt tätäkään kiveä kääntämättä, vaan toi 1976 markkinoille *Outlaw*-räiskinnän. Monien muiden kolikkopelien tapaan Outlaw'sta tehtiin puolivillainen konsolikäännös, mutta hieman yllättäen pelimekaniikka kopioitiin suoraan Gunfightista eikä Atarin omasta pelistä. "Pelisarjaa" lämmiteltiin vielä 2012, kun monien käsien kautta kulkenut Atari julkaisi iLaitteille koskетusnäytön näpyttelyyn perustuvan räiskinnän. Sillä ei tosin ole aiempien osien kanssa juuri muuta tekemistä kuin nimi.

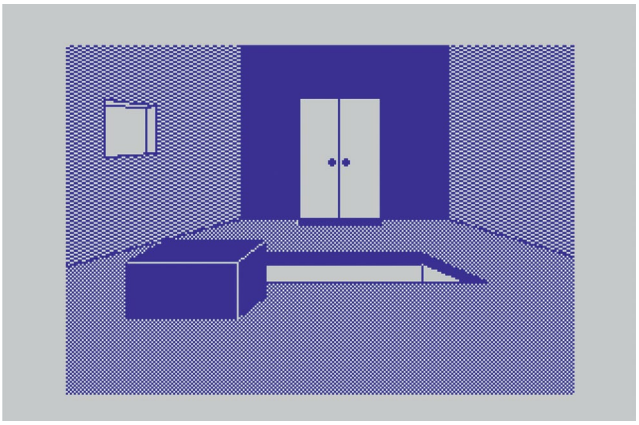
Gunfightin varhaista väkivaltaa.

Kuusnelosella suosittu *High Noon* on sekin selkeä Gunfightin perinteen jatkaja arcademaisine ammuskeluineen, vaikka päähahmo ei olekaan enää jumissa näytön reunassa. Nimen ja tunnusmusiikin osalta ratsastettiin elokuvaklassikko Sheriffin (High Noon, 1952) maineella, vaikka virallinen lisensointi olikin päässyt tekijöiltä unohtumaan, eikä elokuvan uraauurtavasta juonestakaan jäänyt paljoa jäljelle. Varhaisilla kotitietokoneilla nähtiin jonkin verran muitakin ruudinkäryisiä räiskintöjä, kuten *Gunfright*, jossa jahdataan isometrisessä kaupungissa lainsuojattomia, sekä reaktionopeutta testaavat *Bank Panic* -kloonit, joissa täytyy salamannopeasti arvioida, ketä ovistakävelijää sopii ampua.

Ratsastelua ja patsastelua

1980-luvulla kotitietokoneiden länkäritoimintapelit olivat pitkälti tavanomaisen kaavan toisintoja western-kuorrutuksella. Sivuperspektiivi mahdollisti toiminnan kuvaamisen vieritetyssä panoraamassa, johon mahtuivat hevosratsastus, junan katolla hyppely ja monet muut vauhdikkaat kohtaukset. Lännenteemaa onkin usein istutettu aivan perinteisiin tasohyppelyihin – esimerkiksi monilla laitteilla nähdyt *Lucky Luke* -pelit tuntuvat järjestään varsin kaavamaisilta rahastuksilta.

Villin lännen käyttäminen aiheena velvoitti kehittäjiä sisällyttämään peleihin jonkinlaisen ratsastusosuuden sekä pidättäytymään perinteisessä, tunnistettavassa asevalikoimassa revolvereineen ja Winchestereineen. Nämä kaksi seikkaa erottavatkin lännenpelien mekaniikan aikakauden tavallisemmasta



Saman teeman parissa. Sierran Gold Rush! ja ZX Spectrumille tehty Kultakuume.

scifi-räiskinnästä ja ramboilusta. Esimerkiksi pelihallikäännös *Express Raider*, Ultimaten vaatimattomaksi jäänyt *Outlaws* ja budjettipeli *Kane* sisältävät kaikki vauhdikasta ratsastamista. Vaikka toimintapelit tavoittivatkin Hollywood-stunttien tunnelmaa, palikkaisten hahmojen mittelöt olivat kaukana klassisten lännenelokuvien moraalisista ja yhteiskunnallisista pohdinnoista.

Lännenseikkailut

Tekstipohjaiset seikkailupelit toivat lännenteemoihin laajempia ulottuvuuksia räiskinnän ja toiminnan jäädessä taka-alalle. Kaikki seikkailut eivät olleet puhtaasti tekstimuotoisia – esimerkiksi Spectrumin *Wild Bunchia* (ei sukua elokuvalle) ohjattiin monivalintakäskyillä. Wild Bunchissa sankari etsii käsiinsä rikollisjoukkoa etsintäkuulutusten antamien visuaalisten vihjeiden perusteella. Välillä täydennetään tarvikkeita kaupassa, toisinaan tienataan rahaa pokeripöydän ääressä.

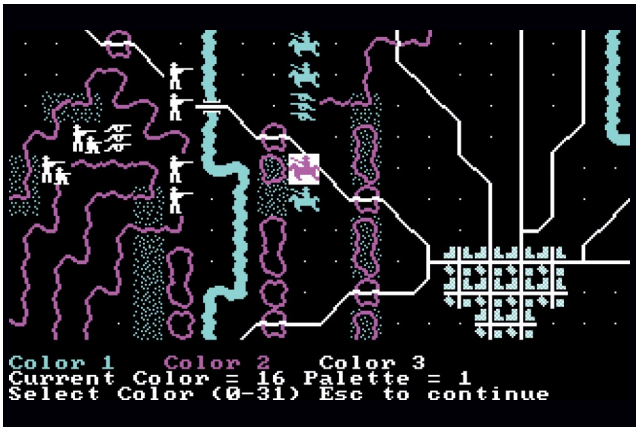
1980-luvun alun uudet konesukupolvet mahdollistivat lännen rikkaan visuaalisen maailman sekä monimutkaisten juonikuvioiden yhdistelyn. Digitaalinen länsi ei ollut enää välttämättä loputon ammuskelukohtaus, vaan tasapainotetumpi seikkailun ja toiminnan yhdistelmä. Commodore 64:n *Law of the West* oli eräänlainen suunnannäyttäjä, vaikkei aivan seikkailupelien sarjaan yltänytkään. Sheriffi-elokuvasta ammentava yksinäinen päähahmo pyrkii ratkaisemaan tilanteita lähinnä monivalintakeskustelujen kautta. Osaavan sanailijan ja aseenheiluttajan käsissä pankkiryöstäjät päätyvät hautaansa, henkipatot luopuvat aseistaan ja naisasiatkin saadaan kuntoon.

Varsinaisista seikkailuista Sierran *Gold Rush!* on pikkutarkaksi yltyvä historiallinen kultaryntäyseepos, jossa päähuomio kohdistuu kultakentille matkustamiseen. Pelistä tulee väistämättä mieleen klassinen *Oregon Trail* -pelisarja, jossa lännen tiellä kohdataan niin ikään vaaroja toisensa jälkeen. *Leisure Suit Larrystä* tunnetun Al Lowen *Freddie Pharkas* taas edustaa huumorilla höystettyä länttä, jossa nälvitään tuttuja kliseitä joka käänteessä. Harvoista kotimaisista seikkailuista mainittakoon tässä vähälle huomiolle jäänyt *Kultakuume*-tekstipeli Spectrumille sekä Sierralle ja LucasArtsille syvään kumartava *Fester Mudd*.

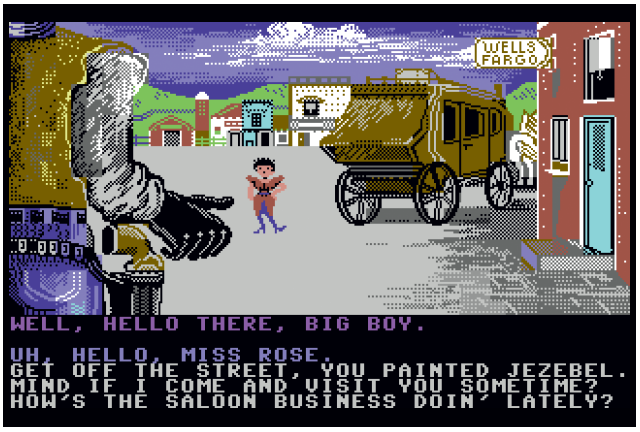
Sisällissodan melskeissä

Yhdysvaltain sisällissota (1861–1865) toimii elokuvissa usein tapahtumien taustana, kuten Sergio Leonen spagettiklassikossa Hyvät, pahat ja rumat. Teema nousee muutenkin esiin vanhoina kaunoina, pohjoinen–etelävastakkainasetteluna sekä hahmojen motiiveja selittävänä tekijänä. Arkaluontoisena aiheena sisällissota on näyttäytynyt peleissä lähes yksinomaan abstraktien ja puolueettomien strategiasimulaatioiden muodossa; etenkään vanhoissa toimintapeleissä sisällissotaa ei edes sivuttu.

Kevyemmistä teoksista nostettakoon esiin kulttipeli *North & South*, joka perustuu belgialaiseen Sinitakit-sarjakuvaan ja edustaa siten Lucky Luken tapaan eurooppalaista näkökulmaa. Strategian raskaammasta päästä löytyy esimerkiksi SSI:n pelisarja, jossa sisällissodan ratkaisevat taistelut Antietamista Gettysburgiin käytiin



Viihdyttävä North & South ja SSI:n karumpi Gettysburg.



Selän takaa kaksi- ja kolmiulotteisesti: Law of the West ja Six-Guns.

jo Apple II:n ja C-64:n valtakaudella. 2000-luvulla sisällissotaa on hyödynnetty muun muassa History Channel -tuotteistettujen fps-räiskintöjen näyttämönä.

Yhdysvalloissa hiljattain leimahtanut kiista Etelävaltojen lipun käytöstä rasistisena symbolina kertoo siitä, kuinka pitkät jäljet 150 vuoden takaisilla tapahtumilla saattaa olla. Riita kosketti pelialaakin, kun alati poliittisesti korrekti Apple poisti App Storestaan kaikki sovellukset, joissa lippu näkyi, mukaan lukien sisällissotaan sijoittuvat strategiapelit.

Ihan kuin elokuvissa

Lännenelokuvia soviteltiin vuorovaikutteiseen pelimuotoon niinkin varhain kuin 1974, kun Nintendo toi pelihalleihin *Wild Gunmanin*, jossa mutkikas kahta projektoria käyttävä mekanismi loihti ruudulle filmiltä aidonnäköisiä vastustajia. Vuorovaikutteisen elokuvan seuraavaa sukupolvea edustivat laserdisc-pelit, joista useimmat tunnistanevat ainakin Cinematronicsin sarjakuvamaisen *Dragon's Lairin* (1983) jatko-osineen.

Seuraavana vuonna laserdisc sai palvella myös Villin lännen tarpeita, kun Konami julkaisi konseptiltaan Dragon's Lairin kanssa varsin samanlaisen reaktiotestin nimeltä *Badlands*. Piirrettyyn western-animaatioon oli japanilaisille peleille ominaiseen tyyliin fuusioitu muitakin fantasiaelementtejä – ainakaan aivan perinteisimpään lännenkuvastoon ihmiseksi naamioituneet hirviöliskot ja kaksipäiset dinosaurukset eivät kuulu.

Massiiviset laserdiscit eivät valloittaneet koteja merkittävästi, mutta niiden perintö näkyy etenkin 1990-luvun huonomaineisissa CD-ROM-peleissä, joissa vuorovaikutteisuus oli samaten typistetty yksinkertaiseen klikkailuun. Tunnetuimpia esimerkkejä formaattiloikan tehneistä peleistä on koominen *Mad Dog McCree*, joka käännettiin alkuperäisestä kolikkopelistä lukuisille CD-pohjaisille konsoleille ja myöhemmin muun muassa Nintendo Wiille. Pelilogiikan yksinkertaisuuden vuoksi pelistä saatiin tehtyä jopa DVD-soittimella toimiva versio, jota ohjaillaan kaukosäätimen nappeja painelemalla

Kohti 3D-länttä

Kolmiulotteinen vektorigrafiikka on monessa mielessä kuin luotu lännenpelejä varten: yksityiskohtaiset maisemat ja ampumiskohtausten vaatima tilallinen hahmottaminen voidaan toteuttaa 3D:n keinoin. Siirtyminen kolmanteen ulottuvuuteen ei tapahtunut kuitenkaan aivan hetkessä. Japanissa julkaistiin 1990-luvun loppupuolella löyhästi lännenteemoista ammentavia toiminta- ja roolipelejä, kuten *Wild Arms* -sarja, mutta puristia tällainen happofuusio ei tyydytä. Selkeämmin Villiin länteen sijoittuva *Outlaws* (1997) oli vielä suhteellisen yksinkertainen *Doom*- ja *Quake*-vaikutteinen ammuskelu, ja vasta myöhemmät *Red Dead Revolver* sekä sen jatko-osa *Redemption* antoivat esimakua siitä, kuinka länttä parhaimmillaan voitaisiin simuloida.

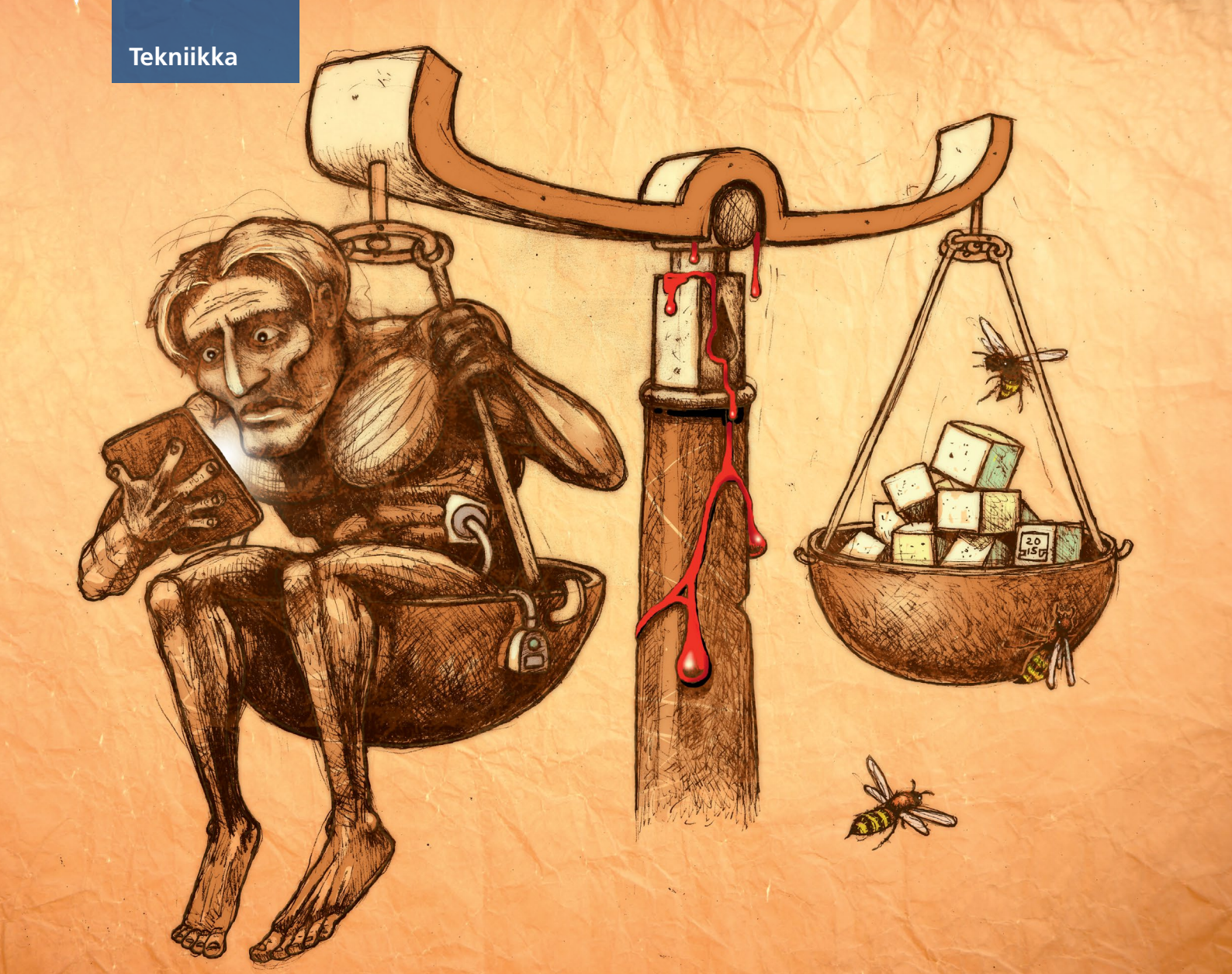
Nykykonsolien ja tietokoneiden 3D-grafiikalla voi luoda rikkaita hiekkalaatikkomaailmoja, joissa pelaajalle tarjoutuu mahdollisuus seikkailla lännessä haluamallaan tavalla. Pelaaja voi osallistua tarinaan – tai halutessaan vain ratsastella ja räiskiä pyssyllä ympäriinsä. Hiekkalaatikkopelit kokoavat yhteen aiemmin erillisiä toimintapelien ja graafisten seikkailujen elementtejä, minkä lisäksi käsikirjoituksista on tullut mittavia ja monisäikeisiä. Siten onkin perusteltua väittää, että lännenpelien genre on noussut uudelle tasolle viimeisimpien teosten myötä.

Aurinko ei laske

Digitaalisten lännenpelien historia yltää vuosikymmenten taakse, ja aihe innoittaa pelintekijöitä edelleen. Ajan saatossa tietyt alalajit, kuten puhtaat kaksintaistelut, ovat menettäneet suosionsa, mutta toisaalta uudet innovaatiot multimediarompuista 3D-grafiikkaan on valjastettu lännenteeman käyttöön. Skaalan kevyessä päässä on paljon sekalaisia pikkupelejä, joissa Villi länsi on lähinnä kuorrutuksena, ja toisessa päässä taas vakavia miljoonatuotantoja.

Tuoretta näkökulmaa kentällä tuovat indie-pelit. *Even Cowgirls Bleed* ja *Calamity Annie* kääntävät perinteisen macho-kuvaston päälaelleen, siinä missä *Gunman Clive* leikittelee sarjakuvateemoilla. Nyt, kun elokuvissa ja etenkin televisiosarjoissa on meneillään western-renessanssi, voidaan todeta, että lännenfiktio on tällä hetkellä taas hyvin voimissaan.

Artikkeli perustuu samojen kirjoittajien tutkimusartikkeliin "Once Upon a Time on the Screen – Wild West in Computer and Video Games", joka julkaistiin WiderScreen-verkkolehden Villi länsi -teemanumerossa 1/2015. Linkki: widerscreen.fi/numerot/2015-1-2/upon-time-screen-wild-west-computer-video-games/



TEKNOLOGIA DIABEETIKON APUNA

Lähes puoli miljoonaa suomalaista sairastaa diabetesta, osa tietämättään. Softa, rauta ja open source mullistavat diabeteksen hoitoa – välillä kapinahengessä.

Teksti: Mika Haulo Kuvat: Tapio Lehtimäki, Mika Haulo

Diabetes on joukko haiman vajaatoiminnan sairauksia, joissa verensokeri kohoaa normaalia korkeammaksi. Diabeteksen hoidossa suuren osa hoitopäätöksistä tekee diabeetikko itse. Diabeetikko seuraa verensokereiden käyttäytymistä ja säätelee niitä lääkityksellä ja ravinnolla, tasapainoillen matalan ja korkean verensokerin välillä. Diabetes on vaarallinen tauti: liian korkeat sokerit kasvattavat elinvaurioiden riskiä, liian matalat voivat viedä tajunnan tai jopa hengen.

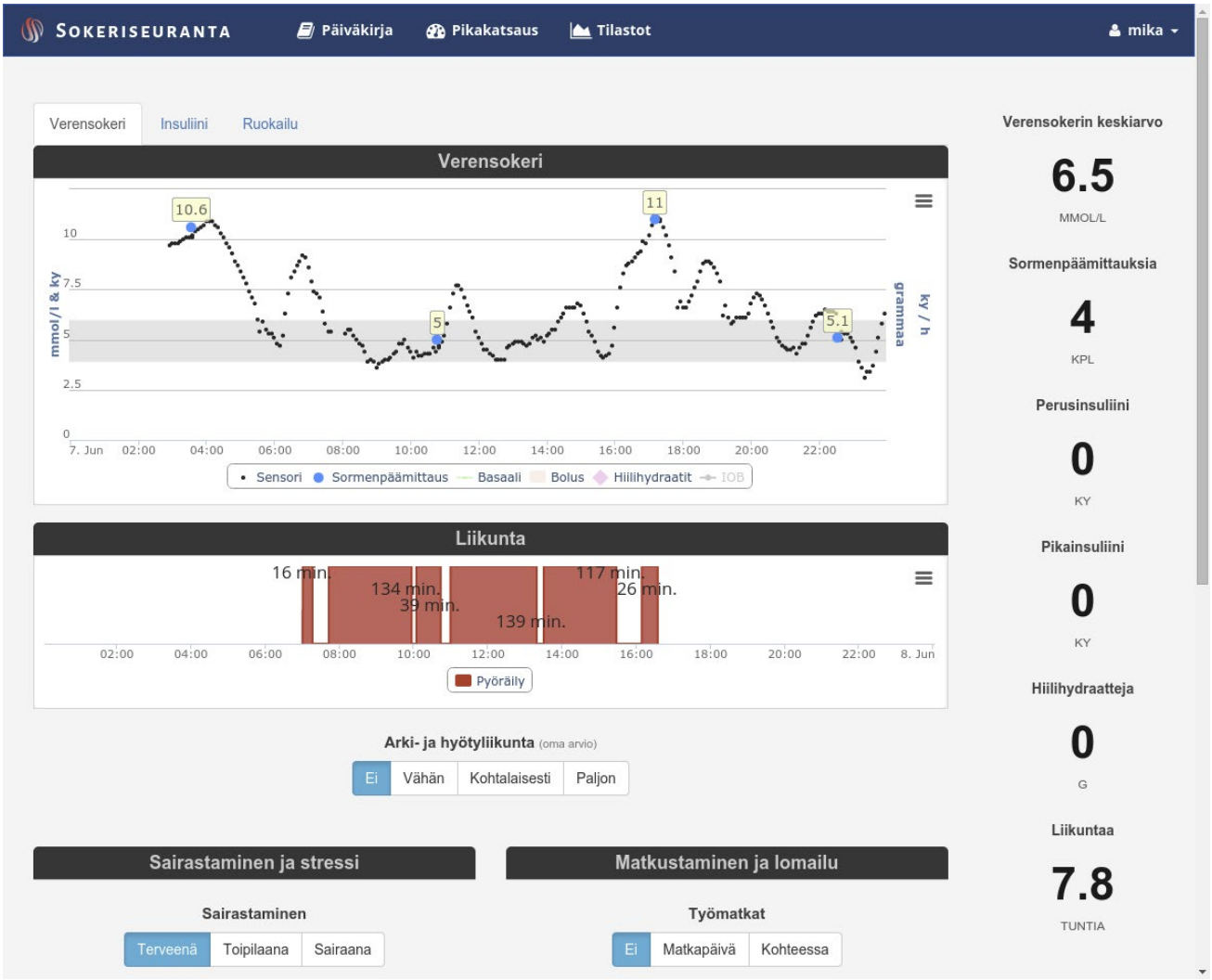
Tässä artikkelissa teemme katsauksen teknologiaan diabeetikoiden oman hoidon apuna. Diabetestyypeistä sanottakoon sen verran, että aktiivisemmin erilaisia omahoidon välineitä käyttävät niin sanotut tyypin 1 diabeetikot, joilla elimistö ei tuota lainkaan verensokeria laskevaa insuliinia, vaan verensokerin säätely on täysin diabeetikon itsensä käsissä. Eniten on tyypin 2 diabeetikoita, he tarvitsevat hoitoa ja seurantaa vähemmän.

Biotekniikka omahoidon perustana

Ilman verensokerin mittausta diabeetikon omahoito olisi kuin autolla ajo silmät kiinni – tuurilla voi pysyä ainakin melkein omalla kaistalla, mutta holtiton eteneminen kostautuu ennemmin tai myöhemmin.

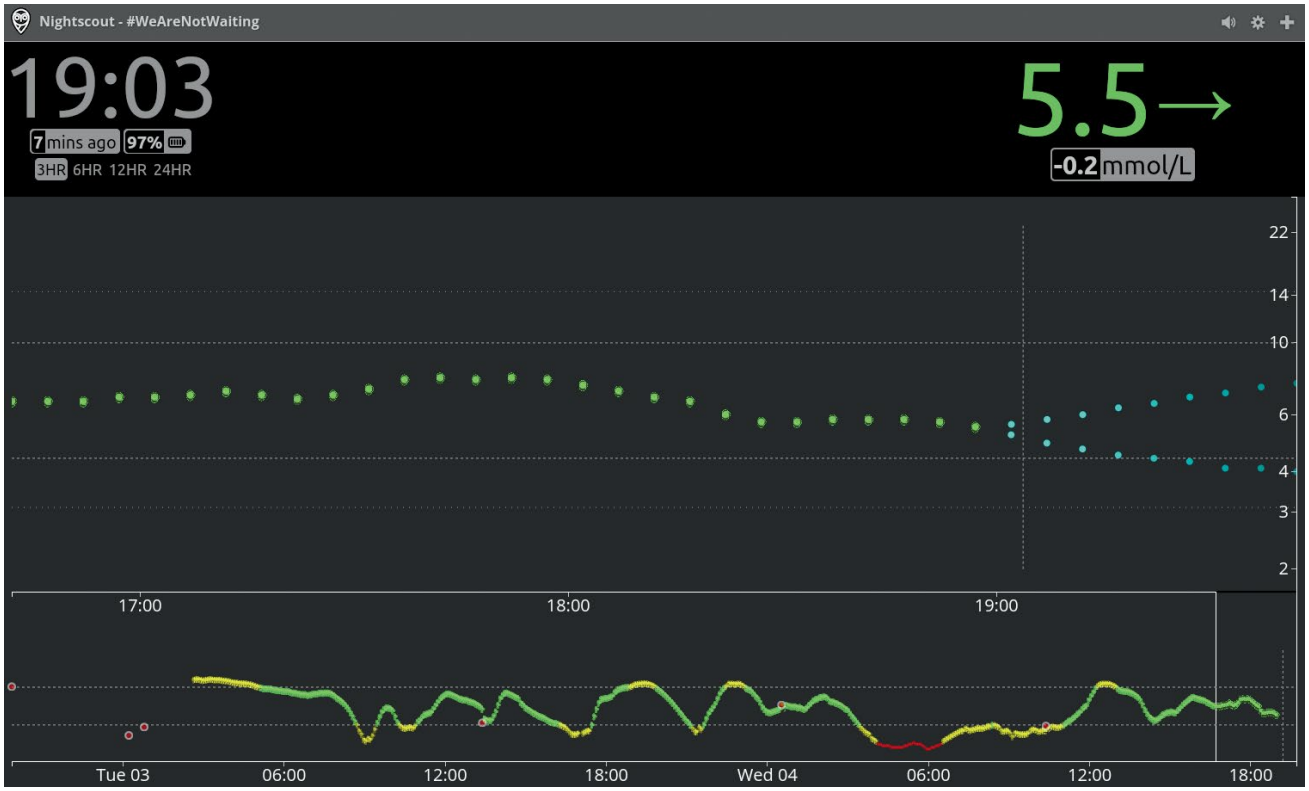
Yksi diabeetikon tärkeimmistä apuvälineistä onkin verensokerimittari kertakäyttöisine mittaliuskoineen. Mittaus perustuu sormenpäästä otettuun verinäytteeseen ja mittaliuskassa tapahtuvaan glukoosioksidaasi- tai glukoosidehydrogenaasientsyymireaktioon – siis yksinkertaisemmin sanottuna sähkövirran muutokseen. Sähkökemiallisesta reaktiosta johdetaan verensokerimittarilla numeroarvo, jonka yksikkö on mmol/l.

Toinen verensokerin seurantaan käytetty laite on glukoosisensori. Se koostuu kolmesta osasta: anturi, lähetin ja vastaanotin. Anturi on ihon alle rasvakudokseen – tyypillisesti vatsan seudulle tai käsivarteen – asettuva ohut lanka, joka mittaa sokeritasoa. Anturiin kiinnitetään lähetin, joka lähettää datan vastaanottimelle. Vastaanotin on pienen mp3-soittimen näköinen laite, jonka näytölle piirtyy muutaman minuutin välein saaduista



Sokeriseuranta-verkkopalvelun päiväkirjanäkymä. Tässä tapauksessa verensokerit ovat pysyneet aisoissa kohtuullisesti – liikuntasuorituksena Pirkan Pyöräily (217 km).

Diabeetikon laukun tyhjennys: Verensokerimittareita, glukoosisensoreita ja insuliinin annostelijoita tarvikkeineen.



Nightscoutin käyttöliittymä on yksinkertaisen tehokas: verensokerin nykyinen arvo, historia muutaman tunnin ajalta sekä arvio verensokerin suunnasta seuraavalle puolelle tunnille.

verensokeriarvoista havainnollinen kuvaaja. Vastaanotin toimii samalla koko sensorijärjestelmän käyttöliittymänä. Nykyään vastaanotin voi olla myös älykännykkä – markkinoilla on sensori, joka lähettää tiedot Bluetooth-yhteydellä.

Insuliinipumppu – uuden ajan taskunauris

Elintärkeää insuliinia voidaan annostella ruiskulla, insuliinikynällä tai -pumpulla. Näistä insuliinipumppu on teknisestä näkökulmasta mielenkiintoisin. Se on mikrotietokone, joka pumppaa säiliöstään insuliinia letkuun ja sitä kautta edelleen diabeetikon kudoksiin. Insuliinia on pystyttävä pumppaamaan ennalta määrätyn ohjelman mukaisesti ja äärimmäisen tarkasti. Tarvittavasta tarkkuudesta kertoo jotain se, että normaalipainoisen aikuisen päivässä tarvitsema insuliinimäärä on vain noin puoli millilitraa. Hyväksyttävän poikkeuksen muodostaa lentokoneen nousu ja lasku, jolloin ilmanpaineen muutokset voivat saada insuliinin liikkeelle diabeetikon haluamatta. Siksi pumpun letku täytyy irrottaa nousun ja laskun ajaksi.

Insuliinin annostelua voidaan ohjelmoida pumppuun monin eri tavoin, esimerkiksi annostelemaan insuliinia aamun ja aamuyön tunteina tavallista enemmän. Tuolloin verensokeri pyrkii tyypillisesti nousemaan. Aterioiden yhteydessä otettava insuliini taas voidaan säätää annosteltavaksi esimerkiksi kahden tunnin aikana. Tällaisesta toiminnosta on hyötyä esimerkiksi pitsaa syödessä, tuhdin aterian verensokeria nostavat hiilihydraatit kun imeytyvät hitaasti.

Tulevaisuuden kehitysaskel on keinohaima – nykyaikainen diabeteksen omahoito pyrkii jäljittelemään terveen haiman toimintaa, visioissa tämä automatisoitaisiin. Teknologia on periaatteessa olemassa: tarvitaan keino tarkkailla verensokeritasoa jatkuvasti (glukoosisensori), väline joko insuliinin tai verensokeria nostavan hormonin glukagonin annosteluun (pumppu) ja keinoäly joka tekee päätöksen verensokeritason säätelystä (minikokoinen tietokone tai älykännykkä).

Vaikka toimivia keinohaimaprototyyppejä on rakennettu, yleiseen käyttöön niitä ei vielä ole odotettavissa. Miljoonien diabeetikoiden erilaiset päivärytmit, ruokailu- ja liikuntatottumukset sekä taudinkuvat ovat haastava yhtälö. Uusimmat insuliinipumput osaavat kuitenkin jo ennakoivasti katkaista insuliinin annostelun tietyissä tilanteissa.

Ohjelmistot kertovat hoidon onnistumisesta

Tyypillisen verensokerimittarin näyttö on pieni yksivärinen LCD. Vaikka verensokerin mittaus on nykyvehkeillä äärimmäisen helppoa, ei mittaustulosten tarkastelu jälkikäteen yksittäisinä numeroina mittarista tai muistivihkosta ole. Siksi tulokset siirretään yhä useammin tietokoneohjelmaan, mobiililaitteeseen tai verkkopalveluun, jossa näyttöpinta-alaa ja laskentatehoa riittää numeroiden havainnollistamiseen.

Lähes kaikilla mittarivalmistajilla on nykyisin jonkinlainen diabetesohjelmisto. Ei kuitenkaan ole muodostunut yhteistä standardia, jonka avulla eri laitteet ja ohjelmistot olisivat keskenään yhteensopivia. Jotkin mittarit keskustelevat kuitenkin Applen HealthKit- ja Samsungin S Health -rajapintojen kanssa. Lisäksi on kolmansien osapuolien ohjelmia, esimerkiksi kotimainen Mendor Balance ja ruotsalainen Diasend, jotka ovat yhteensopivia useiden eri mittareiden kanssa.

Tyypillisellä diabetesohjelmalla voi kirjata muistiin verensokeriarvot, insuliiniannokset, ja aterioiden hiilihydraattimäärät. Monipuolisemmat appsit kirjaavat myös tietoja liikunnasta, mielialasta, pistospaikkojen käytöstä. Tiedot syötetään joko käsin tai luetaan

suoraan laitteista – verensokerimittareista ja -sensoreista tyypillisesti usb- tai usb-serial-yhteyden kautta, joskus saatetaan käyttää infrapunaporttia. Pilvipalveluista voidaan tuoda tietoja REST-rajapintojen ylitse. Myös 3G, NFC ja Bluetooth Smart tekevät tuloaan verensokerimittareihin.

Oma lukunsa ovat hoitohenkilökuntaan yhteydessä olevat mittarit ja palvelut. Sen sijaan että diabeetikko käy vastaanotolla muutaman kuukauden välein, mittaustulokset välittyvät hoitohenkilökunnalle verkossa ja vastaanotolle kutsuttaisiin vain tarpeesta. Tätä on jo pilotoitu.

”Me emme suostu odottamaan”

Terveysteknologiaan liittyy ymmärrettävästi paljon sääntelyä, eikä puolivillaisia viritelmiä lasketa markkinoille, mikä hidastaa kehitystä. Aina käyttävät eivät kuitenkaan suostu odottelemaan validoitujen ratkaisujen ja virallisen terveydenhuollon verkkaista tahtia.

Nightscout on malliesimerkki tällaisesta tee-se-itse-projektista. Yhdysvaltalainen diabeetikon isä päätti koodata yksinkertaisen kännykkäsovelluksen, joka luki tiedot lapsen glukoosisensorista ja lähetti ne verkkopalvelimelle, jonka kautta vanhemmat pystyivät tarkkailemaan lukemia. Parissa vuodessa Nightscout on kasvanut tuhansien perheiden ja diabeetikoiden päivittäiseksi apuvälineeksi, jolle on perustettu jopa säätiö. Projekti on synnyttänyt #WeAreNotWaiting-hashtagin tunnuksena omatoimisuuden ja avoimen lähdekoodin voimalle diabeteksen hoidossa.

Koska Nightscout-systeemi vaatii sensorin vastaanottimen ja kännykän kuljettamista mukana, on sille kehitetty myös koteloita – kuinkas muutenkaan kuin 3D-tulostamalla. Nightscout on myös poikinut muita johdannaisprojekteja: Wixel-piiriin perustuva xDrip sieppaa suoraan glukoosisensorin lähettimen signaalin ja ohittaa virallisen vastaanottimen kokonaan. xDrip on yhteensopiva Nightscout-palvelimen kanssa.

Tee-se-itse-mentaliteetti on edennyt myös keinohaimojen suuntaan, OpenAPS-projektin (Open Artificial Pancreas System) tarkoituksena on luoda avoimen toteutuksen keinohaima.

Kotimainen vastaus huutoon

Suomessakin tapahtuu. Sokeriseuranta-diabetesmuistio on kotimainen tee-se-itse-pilvipalvelu verensokerin kirjaamiseen ja analysointiin. Se erottuu muista diabetessovelluksista irtaantumalla perinteisestä kirjausmallista, jossa verensokeria mitataan vain muutaman kerran päivässä: ennen aterioita ja kaksi tuntia aterioiden jälkeen. Sokeriseurannan lähtökohtana on se, että verensokeria kannattaa seurata huomattavasti tiheämpään.

Pilvipalvelun lisäksi Sokeriseurannasta on julkaistu Android-sovellus, joka osaa lukea dataa verensokerimittareista tiedonsiirtokaapelin avulla. Piuha kiinni mittariin, toinen pää kännykkään ja mittaustiedot tallentuvat verkkoon automaattisesti – muutaman kymmenen koodirivin avulla diabeetikon elämästä on tullut taas hieman helpompaa.

Moderni teknologia mahdollistaa myös uudentyyppisen vertaistuen: Kun joukko suomalaisia ykköstyypin diabeetikoita kyllästyi olemaan diabetes-keskustelun vähemmistössä, he perustivat verkossa toimivan Diabetesseura Plasma ry:n. Yhdistys pyrkii hyödyntämään viimeisintä sosiaalista mediaa ja muuta nykyaikaista tietotekniikkaa nimenomaan tyypin 1 diabeteksen omahoitoon.

Ihmisten apua on toki aikaisemminkin löytynyt verkosta. Perinteikäs Suomen Diabetesliitto ylläpitää omaa Kohtauspaikka-foorumiaan ja Facebookissa on useita diabetesaiheisia vertaistukiryhmiä. Suosituimmassa ykköstyypin diabeetikoille suunnatussa ryhmässä on jo yli tuhat jäsentä ja keskustelu käy vilkkaana päivittäin. Monet diabeetikot kokevatkin saavansa parasta tukea juuri muilta diabeetikoilta.

Hyposprayta odotellessa

Ei-invasiivista verensokerin mittausta, jossa ihoa ei tarvitse puhkaista esimerkiksi verinäytteen ottamista varten, on yritetty jo pitkään saada toimimaan – toistaiseksi ilman kunnollista menestystä. Tuore brittiläinen tulokas Glucosense kehittää laseriin perustuvaa ei-invasiivista verensokerin mittausta. Myös teknologiajätti Google on ryhtynyt etsimään ratkaisuja diabetesongelmiin verensokeria mittaavien piilolinssien muodossa.

Sokeriseurannasta avointa lähdekoodia?

Etsitään tekijöitä mielenkiintoiseen ohjelmistoprojektiin. Ei palkkaa. Paljon työtä. Rivikaupalla dokumentoimatonta koodia. Jatkuva kiire. Jos hanke menestyy, luvassa mainetta ja kunniaa.

Kaikki alkoi artikkelin kirjoittajan omaksi iloksi tekemästä harrasteluprojektista. Kun yhä useampi diabeetikko kiinnostui Sokeriseurannasta, alkoi siitä asteittain tulla vakavammin otettava hanke. Nyt kehitystyö on siinä pisteessä, että yhden miehen resurssit eivät enää riitä kaikkeen tarvittavaan. Palvelinpuolen lisäksi tarvittaisiin pilvipalvelun kanssa juttelevat natiivisovellukset ainakin Androidille ja iOS:lle, ehkä myös Windows Phonelle ja Jollalle. Aika, energia ja taidot eivät yksinkertaisesti riitä tähän kaikkeen – kenties ongelmat ratkeaisivat tekemällä Sokeriseurannasta avoimen lähdekoodin projektin.

Ensimmäisenä kaavaillaan Sokeriseurannan Android-appsin koodin avaamista, tämän artikkelin julkaisun aikoihin. Lisätietoja: www.sokeriseuranta.fi

Diabetesprojekteja

- Nightscout – www.nightscout.info & github.com/nightscout
- Sokeriseuranta – www.sokeriseuranta.fi
- xDrip – stephenblackwasalreadytaken.github.io/xDrip/
- OpenAPS – openaps.org & github.com/openaps/openaps
- iLet Bionic Pancreas – sites.bu.edu/bionicpancreas/
- Diabetesseura Plasma – www.plasma.fi
- Suomalainen keinohaimaprojekti – www.femma5.com

Kalifornian yliopistossa puolestaan tutkitaan eräänlaista tulostettua sensoritatuointia, joka pienellä testiryhmällä on osoittautunut tarkaksi tavaksi mitata verensokeria. Tutkijoiden mukaan väliaikaisen tatuoinnin avulla on mahdollista analysoida solunesteestä käsin muutakin kuin vain sokeriarvoja. Kenties tulevaisuuden diabeetikon tunnistaakin diabetesrannekkeen sijaan viileästä tatskasta.

Internetin hallintomallit

Internet on ainutlaatuinen hallintomallien laboratorio, jossa toimivat vierekkäin diktatuuri, anarkia ja suora demokratia. Miten valta jakautuu netissä ja mikä on tekniikan rooli?

Teksti: Panu Kalliokoski

Kuvat: Tapio Lehtimäki, Mikko Torvinen

Internet lienee edelleen maailman säätelemättömin ja vapain yhteistoiminnan väline, vaikka villien 1990-luvun lopun vuosien jälkeen moni asia onkin muuttunut. Edelleen hyvin pienillä taloudellisilla panostuksilla voi luoda nettiin omia palveluitaan ja julkaista omia viestejään, jotka voivat muuttaa maailmaa – tai sitten vajota netin äärettömään mereen ja unohtua.

Joskus on vaikeaa muistaa, että netissäkin on rajoituksia. Toisaalta myös netin vapaus saatetaan ottaa annettuna ja ymmärtämättä, etteivät netin perustoiminnan takaavat sopimukset ole itsestäänselvyyksiä.

Tässä artikkelissa tarkastelen sitä, miten netin vapaudet, rajoitukset ja valta muodostuvat. Aloitamme perussäännöistä ja netin taustalla olevista sopimuksista. Etenemme yksittäisen palveluntuottajan vapauksiin, pohdimme hiukan niiden rajoituksia ja sitten katsomme, miten verkkoyhteisöt toimivat. Lopuksi palaamme siihen, mitä netin hallinnosta voi oppia.

Tee mitä vain, mihin pystyt

Aloitetaan lähtökohdista. Mitkä ovat ne tärkeät perussäännöt, joihin kaikki muut netin säännöt perustuvat?

Vaikka netin käyttäjällä on paljon vapauksia, niistä vain harvat ovat oikeuksia. Saat esimerkiksi lähettää videon palvelimelle, jos pystyt, mutta toisella käyttäjällä on yhtäläinen vapaus häiriköidä liikennettäsi, tai videopalvelun ylläpitäjällä on oikeus estää pääsysi palvelimelle. Tämä on hyvin erilainen tilanne kuin fyysisessä maailmassa, jossa esimerkiksi matkustusvapautta on tukemassa oikeus vapaaseen liikkuvuuteen: matkantekoasi tahallisesti häiritsevä ihminen voidaan oikeasti panna aisoihin ja saattaa vastuuseen aiheuttamastaan haitasta.

Internet koostuu palvelimista, jotka tarjoavat erilaisia palveluita, ja verkosta, joka välittää liikennettä. Palvelimet ovat yksityinen resurssi, ja niitä hallinnoidaan yksityisesti. Kuka tahansa, jolla ylipäänsä on nettiyhteys ja tietokone, voi julkaista palvelun, koska mikä tahansa netissä oleva kone voi ottaa yhteyden mihin tahansa muuhun julkisessa netissä olevaan koneeseen. On tosin palvelun tarjoajan vastuulla jotenkin saattaa muiden tietoon, missä palvelu on saatavilla ja miten sitä käytetään.

Palvelimet ovat siis lähtökohtaisesti ylläpitäjiensä diktatuureja: palvelimen haltija määrittää, millaisia palveluita se tarjoaa ja miten se reagoi muiden yhteydenottoihin. Netin perusluonne syntyy siitä, että kaikki valta netin sisällöstä on delegoitu sen käyttäjille itselleen. Ero on huomattava verrattuna esimerkiksi radioasemiin, joiden perustamiseksi tarvitaan kalliita laitteita ja lisäksi kansalliselta viranomaiselta lupa lähettää ohjelmaa tietyllä taajuudella. Miten netissä on saatu delegoiduksi näin paljon valtaa?

Yksityisten palveluiden yhteinen verkko

Vaikka palvelut ovat yksityisiä, niitä yhdistävä verkko on yhteinen resurssi. Verkon eri osat ovat yksityisten toimijoiden hallussa, mutta niiden sääntöjä hallitsevat verkkoyhtiöiden väliset sopimukset, maiden väliset sopimukset ja eri maiden kansallinen lainsäädäntö.

Sopimusten olennainen sisältö on, että verkossa pitää kohdella kaikkea (laillista) liikennettä tasavertaisesti. Verkkoyhtiö ei saa lähtökohtaisesti suosia eikä estää liikennettä sisällön perusteella, ei häiritä eikä estää kilpailijoidensa tuotteiden toimintaa ja niin edelleen.[1]

Verkon toiminnassa on muitakin yhteisiä resursseja kuin liikenteenvälitys, jonka hallinto on melko tiukasti pidetty hajautettuna ja kaikille tasapuolisena. Verkko-osoitteet (sellaiset kuin 83.143.218.2), joiden perusteella koneisiin otetaan yhteyttä, jaetaan nettioperaattoreille tarpeen mukaan.[2] Verkkonimet kuten skrolli.fi delegoidaan hierarkkisesti eri toimijoille: esimerkiksi .fi-nimiä jakaa Suomen viestintävirasto, Ficora. Verkon standardit kirjoitetaan ”tiedoksi” RFC-tiedostoihin, ja niitä voi noudattaa, jos huvittaa ja jos haluaa olla yhteensopiva muiden toimijoiden kanssa.[3]

Lukemattomilla järjestelyillä on siis pyritty luomaan yhteinen leikkikenttä, johon voi perustaa pieniä diktatuureja ja jossa kukaan ei pääse liikaa lyömään toista. Miksi ihmeessä näin vapaa järjestelmä?

Syyt ovat sekä ideologisia että käytännöllisiä. Netin luomista ovat toisaalta ohjanneet idealistiset visiot siitä, mitä kaikkea hyvää sillä saadaan aikaan. Toisaalta kyseessä on yksinkertaisesti kilpailuetu: syntyvaiheissaan Internet oli vain yksi monista kilpailevista teknologioista ja se peittosi kaikki muut ennen kaikkea siksi, että siihen oli niin helppoa päästä mukaan.

Tämä netin perusopetus, että jakamalla muille vapauksia tulee itsekin vahvemmaksi, toistuu uudestaan ja uudestaan netin eri osissa, esimerkiksi yksittäisissä palveluissa, päätöselinten toiminnassa ja uusien yhteistyöalustojen suunnittelussa. Palaamme aiheeseen verkkoyhteisöjä käsittelevässä osiossa.

Vapauden rajat

Sinä päätät, mitä teet omalla koneellasi, ja minä päätän, mitä teen omallani. Missä kulkevat vapautemme rajat?

Ensinnäkin, kaikkea ei pysty tekemään. Jos käytän jonkun muun hallinnoimaa palvelua, en pysty lähtökohtaisesti vaikuttamaan mitenkään siihen, miten tuo palvelu toimii ja kehittyy. Yllättävän harvat käyttäjät ovat tietoisia tästä.

Tällä hetkellä voin asetusten avulla kertoa Googlelle, miten tietojani saa käyttää, mutta Google voi lakata noudattamasta asetuksiani milloin vain. Käyttämäni nettipalvelu tai foorumi voi lakata toimimasta, ja tämä voi johtua yhtä hyvin siitä, ettei sen ylläpitäjä jaksa tukea käyttämääni yhteystapaa, kuin siitä, että minulla sattuu olemaan ylläpitäjän mielestä ruma naama, väärät mielipiteet tai kilpaileva tuote.

En myöskään mahda mitään sille, että Facebook sensuroi kaikki ilmiannetut imetyskuvat, enkä sille, että rakkausrunot.fi sensuroi runot, joissa käytetään tuhmia sanoja. Jos olen

[1] Näitä periaatteita kutsutaan yhteisellä nimellä ”net neutrality”, ja ne on dokumentoitu osoitteessa www.fcc.gov/openinternet. Periaatteita on puolustettu raivoisasti viimeksi vuonna 2014: www.savetheinternet.com/net-neutrality-what-you-need-know-now.

[2] Perinteiset IPv4-osoitteet ovat tosin loppuneet jo kesken: elämme jännittäviä aikoja.

[3] Käytännössä koko Internetin tekniset käytännöt on luettavissa RFC-dokumenteista esim. sivulta: www.rfc-editor.org/rfc-index.html.

poliittisesti aktiivinen diktatuurivaltiossa, en voi estää isoja nettifirmoja myymästä keräämiään tietojani maani tiedustelupalvelulle.

Toisaalta tietyt tekoni ovat rajoitettuja siksi, että niistä voi aiheutua minulle haittaa netin ulkopuolella. Yhdysvalloissa ja muissakin länsimaissa on vaarallista perustaa palvelua, johon käyttäjät voivat lähettää tiedostoja toisten ladattaviksi, koska on aina vaara, että tekijänoikeusjärjestöt haastavat ylläpitäjän oikeuteen varoittavana esimerkkinä – ja korvausvaatimukset voivat olla aivan mielikuvituksellisia.

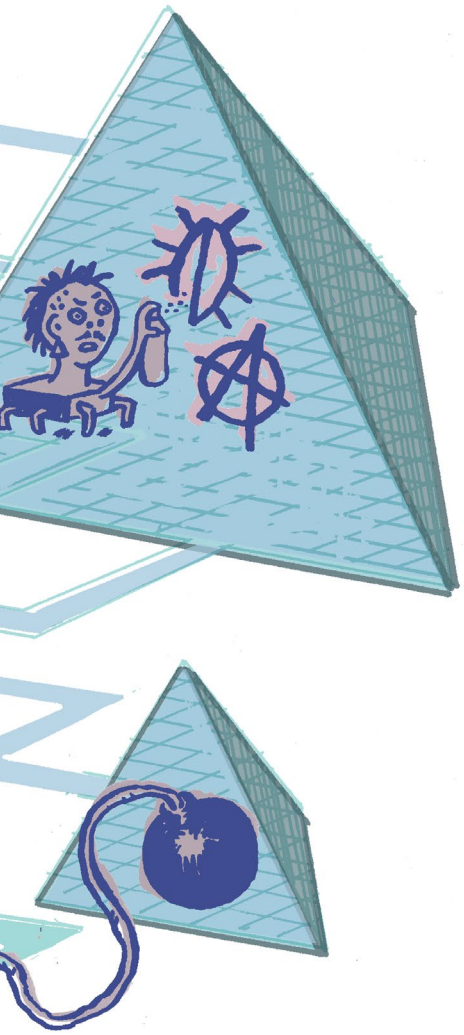
Palveluita voidaan sulkea joko uhkaamalla oikeusjutulla palvelimen tarjoavaa yritystä palvelimen sijaintimaassa tai palvelimen ylläpitäjää tämän oleskelumaassa. Oikeustoimet voivat ulottua myös palvelun sidosryhmiin, koska palvelun ylläpitäjä voidaan pakottaa paljastamaan palvelun käyttäjien tietoja.

Kolmas ja ehkä mielenkiintoisin voima, joka rajoittaa tekemisiäni, on muiden käyttäjien reaktiot. Ihmiset häviävät nopeasti foorumeilta, joita pidetään yllä huonosti, epäoikeudenmukaisesti tai läpinäkymättömästi. Harvat haluavat kehittää avointa ohjelmistoa, jos sen vastuutaho ottaa koodia ja ideoita vastaan mielivaltaisesti tai on muuten hankala. Käyttäjät kaikkoavat palveluista, jotka eivät palvele käyttäjiään, eikä riitaisan työryhmän tehtävänä oleva standardi valmistu koskaan.

Vapaus netissä on siksi hyvin rikkoutuva asia. Niin kauan kuin ei yritä saada mitään aikaan, voi aivan rauhassa olla saamatta mitään aikaan. Kun vakavasti yrittää saada jotain näkyvää aikaan, altistuu erilaisille intressiristiriidoille tai jopa vastahyökkäyksille. Netin tasa-arvoperiaatteet kyllä suojaavat mutteivät rajattomasti.

Diktatuurista anarkiaan

Netissä diktatuurit eivät ole niin pahoja kuin fyysisessä maailmassa, koska ne eivät vie tilaa keneltäkään eikä ke-



nenkään ole pakko olla niiden kanssa tekemisissä. Palveluiden ylläpitäjät kuitenkin rajoittavat usein omaa valtaansa saadakseen palvelulleen käyttäjiä. Käyttäjät ovat pohjimmiltaan palvelun vaikutusvallan mitta: mitä enemmän käyttäjiä palvelulla on, sitä useampi sitä yleensä haluaa käyttää ja sitä vaikeampi sitä on korvata toisella palvelulla.

Ylläpitäjä voi houkutella käyttäjiä palveluunsa esimerkiksi siten, että antaa heille vaikutusmahdollisuuksia palvelun sisältöön. Blogien kommentointimekanismit, tietoa vastaanottavat automaatit, wikit sekä muut kaikkien muokattavissa olevat verkkosivut ovat tästä hyviä esimerkkejä.[4]

Moni käyttäjä luottaa siihen, että kun palvelu on teknisillä keinoilla rakennettu sallivaksi, se myös pysyy sellaisena. Teknisesti avattu diktatuuri on vähän kuin maailma, jossa on yksi jumala (ylläpitäjä) mutta asukkailla vapaa tahto. Näissä maailmoissa on usein kirjoittamattomia sääntöjä, jotka opitaan esimerkin kautta kokeneemmilta käyttäjiltä.

Mutta teknisen avaamisenkin voi tehdä peruuttamattomalla tavalla, jolloin palveluiden hallintomallista tulee anarkistinen. Netin mielenkiintoinen erityispiirre on anarkioiden suuri määrä.

Anarkistiset palvelut on rakennettu siten hajautetuiksi, että ne koostuvat monien osapuolten yhteistoiminnasta eikä kukaan yksittäinen osapuoli pysty tekemään palvelulle muuta kuin lakata osallistumasta siihen. Esimerkiksi webbifoorumeita edeltäneet Usenet-keskusteluryhmät toimivat tällä tavalla: kaikki Usenet-verkostoon kuuluvat palvelimet kopioivat kaikki haluamansa artikkelit muilta Usenet-palvelimilta ja lähettivät vastaavasti muille ne artikkelit, jotka nämä suostuivat ottamaan vastaan. Usenetin luonteen vuoksi keskusteluryhmiin postitusta on hyvin vaikeaa estää ja niiden sisältöä lähes mahdotonta hallita.

Modernimpi esimerkki anarkistisesta palvelusta on kryptovaluutta Bitcoin: valuutan kirjanpito hoidetaan verkkoliikenteellä mutta hajautetulla tavalla niin, ettei yksittäinen toimija pysty helposti häiritsemään järjestelmän toimintaa.

Anarkistisen palvelun toimintatapaa on hyvin vaikeaa muuttaa perustamisvaiheen jälkeen, mikä altistaa sen väärinkäytölle. Jos alkuvaiheessa ei otettu huomioon, miten palvelua voi hyödyntää vääriin tarkoituksiin, palvelun hyväksikäyttö saattaa tehdä siitä alkuperäisille käyttäjilleen lähes käyttökelvottoman.

Tunnetuin esimerkki on tietysti sähköpostiliikenne, josta eri arvioiden mukaan 50–90 % on roskapostia. Sähköposti rakennettiin oletukselle, että kuka tahansa voi lähettää millaisen viestin tahansa. Viestejä ei säädellä mitenkään, joten esimerkiksi viestin lähettäjäksi voi asettaa kenet tahansa. Edelleenkin tämä perusratkaisu vaikeuttaa toivotun sähköpostin erottamista ei-toivotusta.

Anarkistisissakin palveluissa on kuitenkin usein yksi hallinnointitaho, jolla on paljon sanottavaa palvelun kehityssuuntiin: kehittäjät, jotka ovat tehneet palvelun pyörittämiseen tarvittavan ohjelmiston. Kun ohjelmistoa päivitetään, sen kehittäjillä on huomattava valta siihen, mihin suuntaan palvelu kehittyy. Vasta kun palvelun taustaohjelmistoissa on useita keskenään kilpailevia toteutuksia, voidaan puhua tosissaan hajautetusta tai anarkistisesta palvelusta.

Diktatuurista demokratiaan

Eräs tapa yhteishallintoon on käyttöehtosopimuksilla (EULA) muille jaetut oikeudet. Varsinkin kaupalliset palvelut suosivat sopimuksia ja ehtoja, joilla he lupaavat käyttäjille jotain mutta myös rajoittavat omaa oikeudellista vastuutaan, jos käyttäjät suivaantuvat jostakin. Käyttöehtosopimukset tarjoavat yleensä varsinkin maksamattomille asiakkaille hyvin heikot oikeudet, mutta suurin osa netin käyttäjistä ei niistä juuri tiedä tai välitä.

Aidompi tapa osallistaa muita käyttäjiä on ottaa heidät mukaan palvelun todelliseen päätöksentekoon. Tämä fyysisestä maailmasta lainattu toimintamalli saattaa tuntua raskaalta tai byrokraattiselta, mutta nimenomaan netissä sen käynnistäminen on hyvin helppoa. Käytännössä ei tarvitse muuta kuin dokumentoida, miten palvelua hallinnoidaan, ja jaella palvelun ylläpito-oikeuksia muille tahoille. Erilaisista hallintomalleista on kertynyt paljon kokemusta, joten ideoita on runsaasti tarjolla: esimerkiksi äänestykset, seuraajanimitykset tai erilaiset kriteeristöt, mitä erityisoikeutetuilta käyttäjiltä edellytetään.

Samalla kun netti on tehnyt ennenkuulumattoman helpoksi perustaa uusia palveluita, on ensimmäisiä kertoja maailmanhistoriassa mahdollista järjestää äänestyksiä nopeasti, tehokkaasti ja toistettavasti. Vaikka yhteiskunnan käytössä sähköisten äänestysten kokeilut ovat onnistuneet heikosti, verkkoyhteisöissä on onnistuttu luomaan äänestyskäytäntöjä, joita on

[4] Maailman ensimmäinen wiki on osoitteessa c2.com/cgi/wiki, ja toinen samanlainen varhainen kokeilu on everything2.com/.

vaikeaa manipuloida. Debian GNU/Linux -käyttöjärjestelmäprojektissa on noin tuhat ”kansalaista”, joiden äänestyksistä on saatu toistettavia ja koneellisesti käsiteltäviä. Debianin äänestykset eivät kuitenkaan ole nimettömiä, joten niitä ei voi sellaisenaan soveltaa valtioiden käyttöön.

Mukanaolo päätöksenteossa saattaa sitouttaa käyttäjät palveluihin ja niihin perustuviin hankkeisiin. Demokraattisia (kuten anarkistisiakin) yhteisöjä toisaalta vaivaa joskus kiistely ja siitä aiheutuvat sosiaaliset ongelmat. Joskus on helpompaa vain antaa diktaattorin ratkaista ristiriidat parhaaksi näkemällään tavalla.

Toisaalta on vaikeaa olla diktaattori niin, ettei lopulta söisi yhteisöä sisältä, kun vahingossa tai lyhytnäköisyyden vuoksi tehdyt epäoikeudenmukaisuudet alkavat kasaantua. Moni lupaava yhteistyöprojekti on kaatunut vallan keskittymiseen liian harvoille, harvempi siihen, että päätöksentekoprosessit ovat liian raskaita. Netissä asiat tapahtuvat niin nopeasti, että tämän ehtii nähdä elinaikanaan.

Diktatuurista hierarkkisiin delegointeihin

Kun netti alkoi herättää suurten massojen huomion 1990-luvun puolivälissä, alkoi syventyä myös kuilu teknisten ja ei-teknisten käyttäjien välillä. Sama kuilu on olemassa edelleenkin. Vaikka kaikilla käyttäjillä on periaatteessa mahdollisuus perustaa mitä vain palveluita, huima enemmistö netin käyttäjistä ei vain yksinkertaisesti osaa sitä tehdä. Tästä syntyy tarve palveluille, joissa ei-teknisetkin käyttäjät pystyvät luomaan pienoisdiktatuureja omiin tarkoituksiinsa: oma blogi blogipalvelussa, oma maailma nettipelissä, tiedostojen jakamisasetukset pilvipalvelussa jne.

Tässä ei tietenkään ole olennaisesti mitään eroa siihen, että palvelussa on vain annettu käyttäjille teknisiä keinoja vaikuttaa sisältöön tai toimintaan. Joskus kuitenkin palvelu on toteutettu siten, ettei palvelun alkuperäinen tuottaja voi täysin hallita palvelun käyttäjien luomia mikrokosmoksia. Esimerkiksi verkkopeli toimii yleensä siten, että yksi käyttäjistä jakaa pelimaailman omalta koneeltaan, jolloin maailman tiedot todella ovat täysin tuon käyttäjän hallussa. Jos vielä pelin ohjelmisto on avointa lähdekoodia ja julkaistu sallivalla lisenssillä, saattaa koko pelijärjestelmä alkaa elää omaa elämäänsä alkuperäisestä tekijästä riippumattomasti.

Kun käyttäjät saavat peruuttamattoman vallan omiin tuotoksiinsa, se voi osallistaa heidät jopa tehokkaammin kuin pelkkä mukanaolo päätöksenteossa. Sekä suljetut että avoimet alustat tuottavat valtavia ekosysteemeitä, joissa käyttäjät mainostavat ylläpitämiään palveluita, kutsuvat muita käyttäjiä omiin mikromaailmoihinsa ja panevat tolkuttomia määriä työtä ja energiaa tuottaakseen uusia palveluita muiden kehittämillä alustoilla.

Entä sitten?

Netin keskeinen opetus on tähän mennessä ollut se, ettei suuria asioita voi tehdä yksin. Yhteistyö ei kuitenkaan aina tarkoita, että istutaan yhteen pöytään (edes virtuaaliseen pöytään) ja aletaan keskustella asioista. Uusi yhteistyötapa on välineiden ja vaikutusmahdollisuuksien jakaminen ihmisille. Se on sitä, että rakentaa palvelun tai ohjelmiston, dokumentoi sen käyttötavan ja sanoo ihmisille: ”Katsokaa, näin se toimii. Tällaisia asioita voitte tehdä.” Sitten odotellaan, mitä tapahtuu – jos mitään.

Kun tuottaa uusia asioita, alkuvaiheessa tarvitaan innostusta ja kykyä vakuuttaa muut omista visioista, mutta kun hanke kasvaa, tarvitaan muiden osallistamista ja pitkän aikavälin suunnittelua. Tämä näkyy netissä mutta aivan yhtä lailla fyysisen maailman yhteisöissä ja organisaatioissa.

Omakoodia omahoitoon

Diabeteksen kaltainen krooninen sairaus osoittaa, miksi kaikkien pitäisi osata ohjelmoida.
Janne Sirén

Tietotekniikan tasa-arvoistava voima on kiistaton. Digisisällön materiattomuus, sosiaalinen media, avoin lähdekoodi, verkkokauppa, joukkorahoitus ja globaalit tuotantoketjut ovat nostaneet makuukammareiden amatöörit ammattilaisten rinnalle, pienet yksilöt suurten organisaatioiden pelikentille. Useat harvojen etuoikeudet ovat nykyisin massojen mahdollisuuksia.

Sama murros on tapahtunut kroonisten sairauksien omahoidossa sitä mukaa, kun tieto ja tekniikka – etenkin tietotekniikka – ovat kehittyneet.

Epäilyttävä omahoito

Omahoidolla tarkoitetaan potilaan itsensä suorittamia hoitotoimenpiteitä. Diabeteksen tapauksessa tämä tarkoittaa verensokeriarvojen mittaamista sekä lääkitys- ja ravintomäärien vaihtelua. Sen sijaan, että lääkäri määrää annostelun ennalta, potilas säätelee sitä itse. Sanotaan, että tyypin 1 diabeetikot tekevät jopa viisikymmentä terveyspäätöstä joka päivä.

Olen itse ”tyypin 3” diabeetikko – siis diabeetikon omainen. Muistan kuinka 1980-luvulla kotimme ruokapöytä muistutti sairaalalaboratoriota: äitini ruiskuja, ilmassa lentävää insuliinia, lasisia lääkepurkkeja, paksuneulaisia pistolansetteja, isoja veripisaroita ja värikoodattuja muoviliuskoja, joista pitkän odottelun jälkeen tulkattiin summittainen verensokeriarvo.

Niin alkeellisia kuin välineet olivatkin, modernin omahoidon perusteet olivat jo paikoillaan. Vielä niinkin myöhään kuin 1970-luvulla diabeteksen omahoitoon suhtauduttiin kuitenkin osassa lääkärimaailmaa epäilevästi. Kuten monen sairauden kohdalla aikanaan, myös diabeteksen hoidossa suosittiin satunnaisia parantolajaksoja hoitohenkilökunnan valvonnassa. Potilas lähetettiin niin sanotusti vuorille parantumaan taianomaisesti ja saamaan hoito-ohjeet seuraavalle vuodelle.

Jokainen terve laihduttaja tietää, ettei pari viikkoa painonvartijoissa aina auta, kun arki palaa. Asiat ovat vielä huonommin diabeteksen kaltaisen monimutkaisen sairauden hoidossa, josta ei itsehillinnällä selviä. Jos diabetes ilmenisi vain nukkuessa, sen hoito olisi kenties jo automatisoitu, mutta valveilla muun muassa liikunta, ravinto ja stressi vaikuttavat diabeetikon verensokereihin ennalta-arvaamattomasti.

Omaiskoodaajat kunniaan

Omahoitoon kohdistuneet epäilykset olivat silti ymmärrettäviä. Lääketieteellinen hoito on vakava asia, joka vaatii koulutusta. Potilas nähtiin amatöörinä, jolla ei ollut asiaa edes omiin hoitopäätöksiinsä. Ajan mittaan on löydetty tasapaino, jossa diabeetikoista on koulittu sairautensa asiantuntijoita. Kehittyneet hoitovälineet auttavat: insuliinilääkitys virtaa osin automaattisesti ja verensokerit mitataan tarkoilla digimittareilla. Hoidon onnistumista voi analysoida älylaitteilla lähes reaaliajassa.

Hoitovälineiden kehittyminen onkin avannut uuden taistelukentän. Kukaan ei enää vastusta omahoitoa, mutta kamppailu on siirtynyt välineisiin. ”Hyökkäystä” johtavat koodaajat ja puolustuskannalla ovat konservatiiviset lääke- ja diagnostiikkavalmistajat sekä terveysalan jähmeä sääntely. Pioneerina pidetään nuorten diabeetikkojen vanhempien perustamaa Nightscout-projektia. Omaishoitajat ryhtyivät omaiskoodaajiksi ja kytkivät lastensa verensokerisensorit älypuhelimen kautta pilveen. Virallista reittiä tällainen hoitotasapainon ja elämänlaadun mullistanut muutos olisi vienyt vuosia.

Odottelun sijaan #WeAreNotWaiting-hashtagin alla kulkevasta ruohonjuuritason liikkeestä on muodostunut kasvava ekosysteemi, joka ulottuu jo avoimen laitekehityksenkin puolelle. Eikä tämä ole ainoa vastaava hanke. Suomessa omakoodissa on profiloitunut Skrollissakin kirjoittava ohjelmoija, diabeetikko Mika Haulo, joka jakaa verkossa hoitokokemuksiaan ja koodiaan kanssadiabeetikoille.

Diabeteksen lähellä elävät käyttävät kokemuksiaan ja taitojaan hoidon parantamiseen. Siksi kaikkien pitäisi osata koodata ja soveltaa tietotekniikkaa. Ei kaikista ohjelmoijia tai elektroniikkasuunnitelijoita tule, mutta mitä useampi tuntee teknologian tarjoamat mahdollisuudet muuttaa maailmaa, sitä useampi voi muuttaa läheistensä ja kaltaistensa maailmaa. Amatöörit täyttävät auktoriteettien jättämiä aukkoja. He tuntevat arkensa paremmin ja voivat toimia vapaammin. Myös poikkitieteelliset mahdollisuudet kasvavat, kun lääkärit ja tutkijat osaavat ohjelmoida.

Kenties lopullinen kamppailu kroonisia sairauksia vastaan voitetaan, kun useiden tieteenhaarojen ammattilaiset tekevät yhdessä läpimurron. Sitä odotellessa potilaat voivat saada apua hoitoonsa yllättävältä taholta – omaiskoodaajalta.

C++ standardimankelissa

Myös ohjelmointikielille halutaan kansainvälisiä standardeja. Tässä artikkelissa valotamme, millaista työtä C++-kielen standardointi on ja ketkä sitä tekevät. Saamme pientä tuntumaa myös kielen tuleviin ominaisuuksiin.

Teksti: Ville Voutilainen
Kuvat: Mikko Torvinen, Mitol Meerna

C++-ohjelmointikieli alkoi saada merkittävää jalansijaa 1980- ja 1990-lukujen taitteessa. Useat ohjelmistotalot näkivät kielen standardoinnin järkevänä tavoitteena, jotta kieli ja sen työkalut eivät olisi yhden pääkehittäjän varassa. Kansainvälinen standardisoimisjärjestö ISO julkaisi ensimmäisen C++-standardin vuonna 1998.

Nykyään C++:n standardointikomitealla on noin sata aktiivista jäsentä ja pyrkimyksenä on julkaista uusia standardeja kolmen vuoden välein. Seuraavat ovat suunnitteilla vuosille 2017 ja 2020.

Standardiprosessi

ISO:n hierarkiassa itse standardointityöryhmä eli -komitea on työryhmä, jonka jäseninä on kansallisten standardointiorganisaatioiden nimeämiä asiantuntijoita. Työryhmiä on useille eri kielille, kuten WG21-ryhmä C++:lle, WG14 C:lle, WG4 Cobolille ja WG5 Fortranille. Työryhmät kuuluvat alikomiteaan nimeltä SC22, joka standardoi ohjelmointikieliä. SC22 puolestaan kuuluu JTC1:een, joka on ISO:n ja IETF:n yhteinen komitea informaatioteknologian kehittämiseen. Itse standardointityöryhmän näkökulmasta SC22 ja JTC1 ovat jokseenkin etäisiä katto-organisaatioita, joiden kanssa ei olla paljonkaan tekemisissä.

Standardi syntyy laajennus- ja korjausehdotusten pohjalta. Standardin kehitysvaiheessa standardointikomitealle lähetetään ehdotuksia, jotka komitea käy läpi kokouksissaan. Kun ehdotukset hyväksytään, ne liitetään osaksi standardin työluonnosta, ja kun komitea katsoo olevansa valmis aloittamaan uuden standardin julkaisuvaiheen, työluonnos lähetetään lausuntokierrokselle.

Lausuntokierros kestää muutaman kuukauden, joiden aikana kansalliset organisaatiot lähettävät kommenttinsa komitealle. Kierroksen loputtua komitea kerää annetut kommentit yhteen ja käsittelee ne samaan tapaan kuin muutosehdotukset, tosin sillä erotuksella, että kommentteihin on annettava jonkinlainen vastaus, myönteinen tai kielteinen.

Luonnoksia ja lausuntokierroksia voi tehdä useampia, mutta standardi on mahdollista julkaista yhdellä lausuntokierroksella ja erillisellä – ja samalla lopullisella – hyväksyntäkierroksella, jonka tuloksena ei anneta kommentteja vaan pelkkä hyväksyntä-äänestys. Tähän mennessä kaikki C++-standardien hyväksyntä-äänestykset ovat olleet yksimielisiä.

Komitea koostuu pääasiassa C++-kääntäjien ja -standardikirjastojen toimittajista sekä C++:aa käyttävien yritysten edustajista, mutta kuka tahansa kansallisen organisaation nimeämä asiantuntija voi olla komitean jäsen. Kansallisten organisaatioiden säännöt vaihtelevat maittain: siinä missä jotkut maat vaativat edustajiltaan jäsenmaksun, muissa maissa osallistumiseen ei välttämättä ole erityisempiä ehtoja.

Standardointikomitea kokoontuu varsinaiseen kokoukseen kaksi tai kolme kertaa vuodessa. Kokouksissa komitean jäsenet käsittelevät ehdotuksia ja mahdollisia lausuntokierroksen kommentteja. Kokous kestää yleensä kuusi päivää, ja päivät ovat täyden työpäivän mittaisia. Varsinaisten kokous-

ten ulkopuolella komitea työskentelee myös puhelinkonferenssien ja sähköpostilistojen avulla.

Komitea jakaantuu alityöryhmiin, jotka ovat erikoistuneet tiettyyn alueeseen. Evolution Working Group (jota allekirjoittanut johtaa) vastaa kielen ominaisuuksien suunnittelunäkökulmasta, ja Core Working Group vastaa kielen ominaisuuksien yhdenmukaisuudesta ja määrittelyn lopullisesta sanamuodosta. Library Evolution Working Group vastaa standardikirjaston ominaisuuksien suunnittelunäkökulmasta ja Library Working Group standardikirjaston ominaisuuksien yhdenmukaisuudesta ja määrittelyn lopullisesta sanamuodosta.

Alityöryhmien lisäksi komitean alla on kymmenkunta tutkimusryhmää, joiden vastuulla ovat erilaiset tekniset yksityiskohdat kuten säikeet ja muut rinnakkaisuusominaisuudet (SG1), transaktionaalinen muisti (SG5) ja peliteollisuuden ja pienen latenssin käyttötapausten tarpeet (SG14). Tutkimusryhmien tarkoitus on kerätä tietyn alueen asiantuntijoita myös komitean ulkopuolelta tuottamaan muutosehdotuksia neljän yllä mainitun ryhmän katselmoitavaksi. Tutkimusryhmien toiminta on vähemmän byrokraattista kuin standardointikomitean.

C++-standardointi Suomessa

Ohjelmointikielten standardointiin keskittyvässä SC22-organisaatiossa on sekä osallistujajäseniä että tarkkailijajäseniä. Suomi on ollut SC22:n osallistujajäsen vuodesta 2009 lähtien, jolloin itse liityin C++-komitean jäseneksi. Jäsenyys lähti liikkeelle siitä, kun huomasin komitean olevan aktiivinen ja halusin kysyä tiettyjen ehdotusten tekijöiltä täsmennyksiä ja ehdottaa muutoksia niihin.

Ehdotusten tekijät totesivat, että lausuntokierros on käynnissä ja että minun pitäisi ottaa yhteyttä kansalliseen organisaatiooni, jos haluan muutoksia. Suomen standardisoimisliitto SFS ry. suhtautui myötämielisesti osallistumiseen, ja olen käynyt kaikissa standardointikomitean kokouksissa vuoden 2009 helmikuusta vuoden 2015 lokakuussa pidettyyn viimeisimpään kokoukseen asti. Vuoden 2012 lopusta alkaen komiteassa on ollut kaksi suomalaista edustajaa. Osallistumme komitean kaikkiin kokouksiin, standardiehdotusten laadintaan ja niiden katselmointiin sekä jonkin verran myös toteutustyöhön GCC-kääntäjän ja sen C++-kirjaston, libstdc++:n osalta.

Suomen asiantuntijaedustus ei vaadi edustajilta jäsenmaksua, ja muutenkin kansallinen strategia tukee informaatioteknologiastandardeja. Kokouksista tulee kustannuksia matkakulujen ja majoituksen muodossa, minkä vuoksi on tärkeää, että edustajien työnantajat tukevat standardointityötä myös rahallisesti. Oma työnantajani Symbio maksaa kokouksiin liittyvät menot ja järjestää standardointikomitean kokouksen Oulussa kesällä 2016.

Standardoinnin motivaatio

Miksi standardoida ohjelmointikieli, kun monet kielet kehittyvät ilmankin näin raskasta prosessia? C++ standardoitiin alun perin siksi, että useampi kaupallinen toimittaja halusi yhteisen määrittelyn. Teollisten käyttäjien näkökulmasta standardointi parantaa siirrettävyyttä erilaisten toteutusten välillä ja luo vakautta: kielen ja standardikirjaston määrittely ei ole minkään yksittäisen tahon sanelun tulos.

Vuoden 2011 standardi toi mukanaan suuren määrän merkittäviä parannuksia, ja niistä yksi merkittävä on säikeiden tuki. Samoihin aikoihin monet kaupalliset ja teolliset käyttäjät alkoivat kiinnittää huomiota virrankulutukseen sekä mobiilialustoilla että konesaliympäristöissä.

C++:n merkittävänä etuna on perinteisesti ollut suorituskyky ja suurten, monimutkaisten ohjelmistojen hallinta. Kielellä voidaan kirjoittaa hyvinkin abstraktia koodia ilman suorituskykyongelmia, ja yhteinen standardi ja toimittajariippumattomuus johtavat käytännössä hyvään siirrettävyyteen eri alustojen välillä.

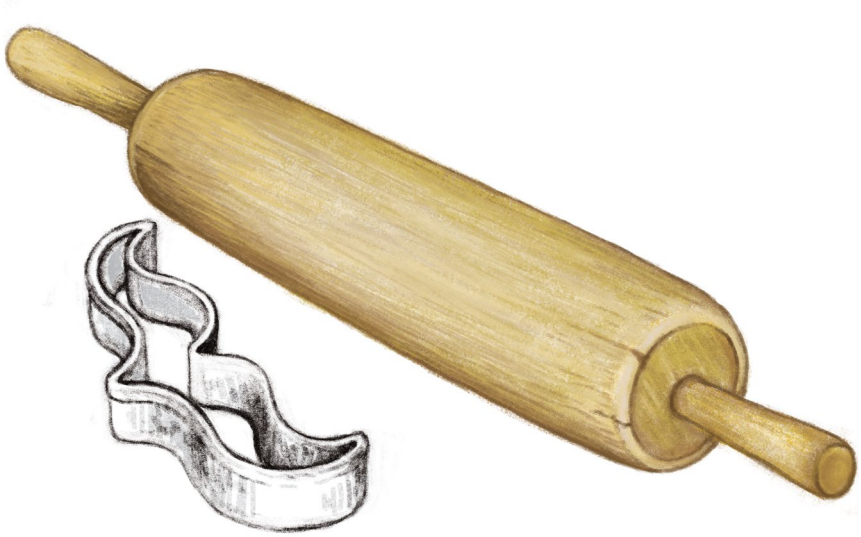
Monilla yhtiöillä kuten Microsoftilla, Intelillä, IBM:llä ja Googlella on perinteisesti ollut melko pieni asiantuntijaedustus standardointikomiteassa, mutta viime aikoina yhtiöt ovat lisänneet osallistumistaan. Vuoden 2011 standardin myötä kielen suosio näyttää kasvaneen merkittävästi.

Tulevat standardit

Vuoden 2011 standardi oli massiivinen muutos, mutta sen jälkeiset standardit tulevat luultavasti olemaan muutosmäärältään pienempiä. Esimerkiksi vuoden 2014 standardi toi tarkoituksellisesti vain pieniä muutoksia, joilla pyrittiin paikkaamaan edellisen version puutteita.

Vuonna 2017 julkaistavaan standardiin on taas tarkoitus lisätä enemmän uusia ominaisuuksia. Sen lopullinen sisältö ratkeaa kevään 2016 aikana. Vuonna 2020 julkaistavassa standardissa on jälleen pyrkimyksenä lähinnä vain korjata vuoden 2017 standardin puutteita. Komitea pyrkii tällaiseen julkaisurytmiin, jossa isompia muutoksia sisältävät standardit ja pienemmät korjausversiot vuorottelevat, siten että uusi standardi julkaistaan aina kolmen vuoden välein.

Vuodesta 2014 lähtien komitea on hyödyntänyt enenevässä määrin mahdollisuutta julkaista niin kutsuttuja teknisiä määrittelyjä. Nämä ovat



määrittelyjä, joita voidaan julkaista standardien rinnalle nopeutetulla aikataululla. Määrittelyjä on jo julkaistu transaktionaaliselle muistille, konsepteille, tiedostojärjestelmätuelle, kirjastotyypeille ja rinnakkaisuustoiminnoille. Määrittelyjä pyritään tuomaan varsinaiseen standardiin sitä mukaa, kun niistä saadaan toteuttajien ja käyttäjien palautetta.

Vuoden 2017 standardin sisältö on, kuten todettu, vielä päättämättä. Viimeisimmässä kokouksessa kuitenkin päätettiin tehdä moduuleista tekninen määrittely. Sen sijaan konseptien kohtalosta ei ole vielä päätöstä.

Muista merkittävistä ehdotuksista vuorottaisrutiinit (coroutine) ovat edenneet varsin pitkälle standardiprosessissa, siinä missä sopimuspohjaisen ohjelmoinnin tuki on keskeneräisempi eikä välttämättä päädy tähän standardikierrokseen. Kirjastoehdotuksista range-pohjaiset algoritmit ovat päätymässä tekniseen määrittelyyn, kuten verkkotukikin. Edellisten lisäksi komitea käsittelee vielä monia pienempiä ominaisuuksia kuten funktiokutsujen yhdenmukaistamista, pisteoperaattorin uudelleenmäärittelyä ja automaattista vertailuoperaattorien määrittelyä.

Komitean aikataulutavoite on julkaista standardiluonnos lausuntokierrokselle kesän 2016 Oulun-kokouksen jälkeen. Lausuntokierrosten kommenttien käsittelylle varattaneen aikaa muutama kokous, minkä jälkeen standardi on tarkoitus viimeistellä niin, että se voidaan lopullisen äänestyksen jälkeen julkaista vuoden 2017 aikana. Kun standardi lähtee viimeiselle äänestyskierrokselle, seuraavan standardin sisällön laadinta on jo hyvässä vauhdissa ja tähtää vuoden 2020 standardiin.

Tällaista on suunnitteilla

Konseptit: Tyyppimallit templateille (malleille), joiden avulla voidaan määrittää kategorisesti, mitä malli vaatii sille annetuilta tyypeiltä. Niiden avulla malleista saadaan merkittävästi parempia virheilmoituksia silloin, kun vaatimukset eivät täyty.
Alkuperäinen kuvaus: open-std.org/JTC1/SC22/WG21/docs/papers/2013/n3580.pdf.
Uusin luonnos: open-std.org/JTC1/SC22/WG21/docs/papers/2015/p0121r0.pdf.

Vuorottaisrutiinit (coroutine): Funktio, jonka suoritus pystytään pysäyttämään ja suoritusta jatkamaan myöhemmin.
Alkuperäinen kuvaus: open-std.org/JTC1/SC22/WG21/docs/papers/2014/n4286.pdf.
Uusin ehdotus: open-std.org/JTC1/SC22/WG21/docs/papers/2015/p0057r0.pdf.

Range-algoritmit: Algoritmeja ja konsepteja, joiden pohjalta voidaan yhdistellä operaatioita yhtenäisen ohjelmointimallin mukaan ja kirjoittaa koodia, joka prosessoi dataa ilman silmukoita ja ehtolauseita, häviöttömästi. Alkuperäinen kuvaus:
open-std.org/JTC1/SC22/WG21/docs/papers/2014/n4128.html.
Uusin ehdotus: open-std.org/JTC1/SC22/WG21/docs/papers/2015/p0021r0.pdf.

Transaktionaalinen muisti: Ohjelmointitekniikka, jonka avulla voidaan kirjoittaa rinnakkain suoritettavaa koodia ilman käsin kirjoitettuja tahdistusoperaatioita.
Alkuperäinen kuvaus: open-std.org/JTC1/SC22/WG21/docs/papers/2013/n3718.pdf.
Uusin ehdotus: open-std.org/JTC1/SC22/WG21/docs/papers/2014/n4301.pdf.

Linkkejä

- Standardien yleiskuvaukset: en.wikipedia.org/wiki/C%2B%2B11 (C++11), en.wikipedia.org/wiki/C%2B%2B14 (C++14), en.wikipedia.org/wiki/C%2B%2B17 (C++17).
- Pääsivusto kaikelle standardiin liittyvälle: isocpp.org.
- Komitean työsivut, jotka sisältävät kaikki ehdotukset ja niiden aikaisemmat versiot: open-std.org/JTC1/SC22/WG21/.
- Video konsepteista: www.youtu.be/qwXq5MqY2ZA, www.youtu.be/NZeTAnW5LL0.
- Video range-algoritmeista: www.youtu.be/mFUXNMfaciE.
- Video vuorottaisrutiineista: www.youtu.be/_fu0gx-xseY.

Hei,

Olen täysin ällikällä lyöty. Miten voi olla mahdollista, että on olemassa näin täydellinen tietokonelehti?

Olen raskain mielin seurannut perinteisten lehtien loppumattoman tuntuista alamäkeä:

- Yksi muuttui automarkettien tuotekatalogiksi
- Toisesta tehtiin naistenlehtien tyyppinen henkilöhaastatteluläpyskä
- Kolmas tapettiin koska jollain talouslehdellä oli yhteneväistä lukijakuntaa

Nyt kädessäni on taivaasta tipahtanut lehti, johon on poimittu kaikkien kolmen parasta aineistoa.

Kiitos toimitus, kiitos! Jatkakaa mahtavaa työtä ja eläköön hämmästyttävä lehtenne ikuisesti.

Nimim. ”Tippa linssissä”

Kyberrikosten testiympäristö

Suomi pyrkii kyberturvallisuuden kärkimaaksi. Pieni maa saattaisi päästä tavoitteeseen saranapuolelta. Kyberrikosten suorittamisesta pitäisi tehdä kansalaistaito ohjelmoinnin tapaan. Kun osaa rikoksen alkeet, ymmärtää myös riskit ja suojautumisen tarpeen sekä osaa varautua uusiin tekotapoihin. Varsinkin asioiden internetin eteneminen voi muuten kasvattaa rikosavaruuden hallitsemattomaksi.

Alan yrityksillä ja oppilaitoksilla on suljettuja testiympäristöjä ja laboratorioita. Vastaavia testiympäristöjä pitäisi perustaa avoimesti kaikkien kokeiltavaksi. Selvästi tunnistettava testiportaali, jonka alta löytyvät testattavat järjestelmät. Fyysisiä laitteita kuten maksupäätteitä voisi hakkeroida valvotuissa julkisissa tiloissa. Yleisesti tunnetut hyökkäysmenetelmät ja -työkalut ohjeineen julkistetaan. Testiportaalin ympärille voisi rakentaa alan palveluja ja muuta liiketoimintaa kattavasti.

Testiympäristössä ei ole lainkaan liikennettä, joten kaikki aktiivisuus on hyökkäystä ja siten helpompi havaita. Jos tarvitaan testiliikennettä, sen simuloi robotti ja se on helppo suodattaa pois muusta liikenteestä. Hyökkäys voidaan kohdistaa halutulle alueelle antamalla haastetehtäviä. Onnistunut hyökkäys voi vaatia uuden menetelmän tai työkalun kehittämistä sekä jälkien peittämistä. Erityisesti näiden luulisi kiinnostavan alan yrityksiä.

Toiminnan on oltava avointa. Testirikosten kohteeksi joutuneet yritykset julkistavat tapauksia ja palkitsevat tekijöitä näkyvästi. Luottamus yritykseen ja sen järjestelmiin kasvaa avoimuuden myötä. Kiinnostus hyökätä yrityksen oikeaan järjestelmään laskee ja rikolliset valitsevat vähemmän testattuja järjestelmiä. Hyökkäyksistä vaikenevaa yritystä aletaan pitää epäluotettavana. Ehdottoman varmoihin järjestelmiin uskoo tulevaisuudessa yhä harvempi.

Testiportaali on yleisessä verkossa. En tarkoita virusten ja muiden saastuttavien haittaohjelmien ohjelmoinnin opettamista enkä käyttöä. Vertaan ajatustani tietokonepeleissä huijaamiseen. Systeemiä mitenkään vahingoittamatta saadaan sovellus toimimaan ennakoimattomalla tavalla. Juuri näitä temppuja haen.

Esimerkiksi otan Bitcoinin murtamisen. En voi mitenkään uskoa, että se on mahdotonta. Valuuttaa ei ehkä voi varastaa itselleen, mutta varmennuskirjaukset pystyy varmaan sekoittamaan. Nehän ovat avoimesti verkossa, tosin salattuina. Varmennuksista tulee heti mieleen analogia dna-koodiin tai kvanttiteoriaan. Poikkitieteellinen asiantuntijaryhmä saattaisi Bitcoinin kaataa. Vastaavan virtuaalivaluutan tai muun varmenteen kehittäjä voisi laittaa testin avoimesti verkkoon, koska tosiaineistokin olisi siellä kaikkien nähtävillä.

Lopuksi pahan ajatukseni muusa eli asioiden internet. Mitä tapahtuu, kun yhä enemmän laitteita kommunikoi keskenään verkossa? Valmistajat ovat sekalaista seurakuntaa. Hintakilpailun paineessa turvallisuuteen ei voida panostaa riittävästi. Ja kuka osaa ennakoida eri laitteiden yhteisvaikutuksia? Välttämättä ei tarvita rikollisiakaan asioita sotkemaan, ongelmia voi ilmetä riittävästi ilmankin.

Parhain terveisin Uolevi Kattun

Korjauksia aiempien numeroiden paperiversioihin

2015.1

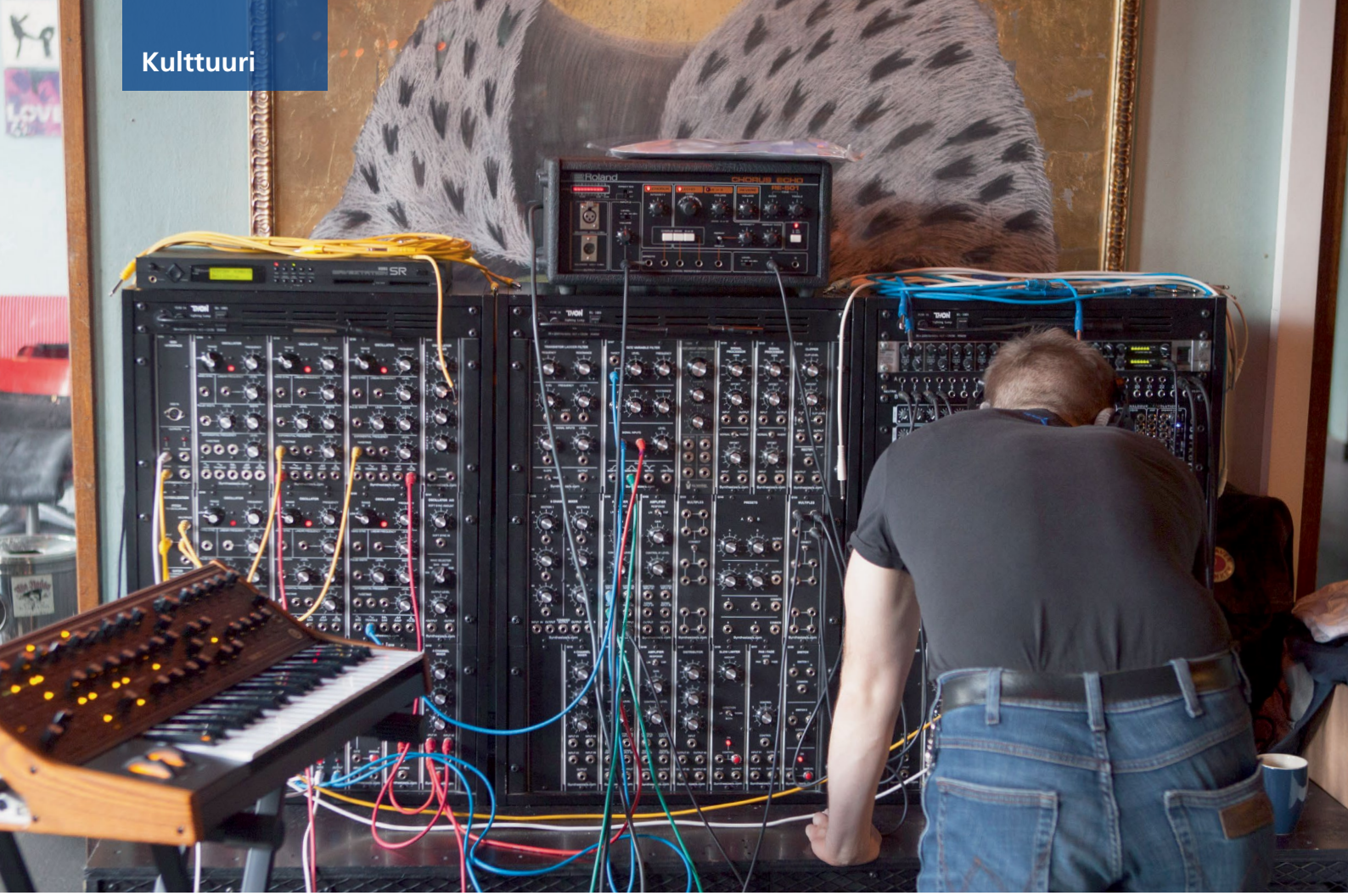
- Rust-juttuun sivulle 37 eksyi vahingossa sama koodilistaus tuplana. Oikean koodilistauksen löydät Skrollin pdf-versiosta, osoitteesta skrolli.fi/2015.1.graffiti.pdf.

2015.2

- Numerosta jäi uupumaan sisällysluettelon taustakuvan tekijätieto. Loistavan valokuvan meille nappasi Jari Viitala.

2015.3

- Sivulta 26 alkavasta Vim-jutusta uupui Vimin tekijän, Bram Moolenarin nimi.



Turku – Suomen syntetisaattoripääkaupunki

Suomessa syntetisaattoreita on toki harrastettu jo pitkään, mutta isompia synaintoilijoiden kokoontumisia on alettu järjestää vasta viime vuosina. Turussa alun saaneen syntetisaattoriseuran vaikutusalue ulottuu jo ulkomaille asti. Mitä suomalaisessa syntetisaattoriskenessä tapahtuu nykypäivänä?

Teksti ja kuvat: Nasu Mäki-Petäys

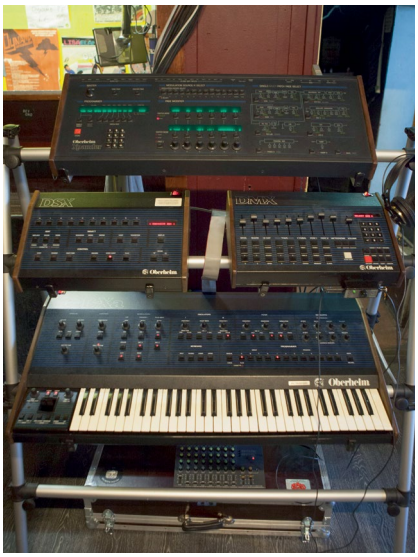
Vuosi 2010 – ruuhkainen baari Turussa. Neljä toisensa tavalla tai toisella tuntevaa muusikkoa istuutuvat saman pöydän ääreen. Kaikki heistä harrastavat syntetisaattoreita.

Illan keskusteluiden seurauksena syntyy idea Facebookiin perustettavasta syntetisaattoriaiheisesta ryhmästä. Se saa nimekseen Turun syntetisaattoriseura, ja sen jäsenmäärä kasvaa nopeaa tahtia yhä uusien ihmisten löytäessä pitkään puuttuneen foorumin harrastuksestaan keskustelemiselle. Kiinnostuneita on kuitenkin ympäri Suomea, ja pian Turun syntetisaattoriseura kattaa nimestään huolimatta jäseniä koko maan laajuudelta. Tänä vuonna jäseniä on jo reilu tuhat – myös ulkomaalaisia! – ja ryhmään kuuluu niin maan kovimpia harrastajia kuin aloittelijoitakin.

Ryhmän tapaamiset kasvavat vuonna 2013 Synamiitti-nimellä tunnetuksi tapahtumaksi, johon jäsenet tuovat keräilemiään harvinaisuuksia sekä itse rakentamiaan projekteja. Myös maahantuojat sekä kaupalliset valmistajat ovat läsnä tapahtumassa.

Suuren jäsenmäärän seurauksena ryhmän viestitulva kohisee kysymyksiä, omatekoisten projektien esittelyitä, kilpailuja, ja toisinaan keskustelu redusoituu huumoriin soittimien äänenlaadun parantamisesta puupäätyjen myötä. Tietomäärä on valtava, ja kinkkisempiin teknisiinkin kysymyksiin saa äkkiä vastauksia, mutta menneitten ohjeitten löytäminen viestitulvasta voi olla myöhemmin niin työlästä, että on helpompi kysellä uudestaan.

Vuoden aikana ryhmään voi tulla useita väliaikaraportteja jonkun itse tekemästä projektista, joista käydään teknistä keskustelua. Useamman eri tason harrastajat oppivat toisiltaan.



Aina ei harvinaisuuksien saaminen paikalle onnistu täydellisesti. Tarjolla ollut Oberheim OB-Xa vikaantui varhain jättäen yhden koskettimista pohjaan tapahtuman loppuajaksi. Myöskään sopivaa kaapeliratkaisua ei löytynyt, joten OB-Xa ohjasi Xpanderia vain joka kuudennella nuotilla.

Harvinaista, uutta ja uniikkia

Jo ensimmäinen synamiitti sai mukavasti mediahuomiota. Paikalla oli harvinaisuuksia kuten Mellotron, muita merkittäviä laitteita kuten omatekoinen Roland TR-808 -klooni ja peilipintainen Shruthi-syntetisaattorimalli, joka perustuu avoimen lähdekoodin ohjelmistoon ja avoimeen rautaan. Nokialainen putkielektroniikkavalmistaja Verde Audio oli tapahtumassa esittelemässä tuotantoaan. Vierailijoita kävi noin 100–150 päivän aikana.

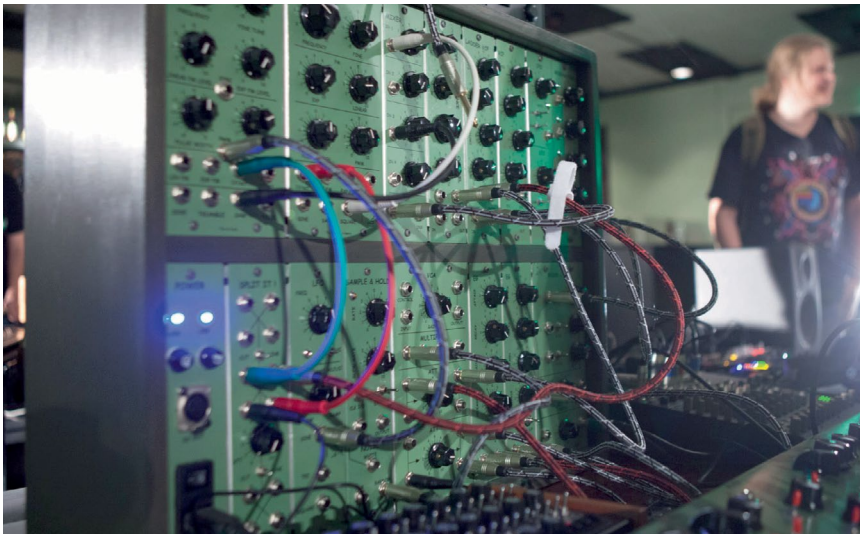
Tänä vuonna tapahtuma oli kasvanut kaksikerroksiseksi, josta alempaan kerrokseen oli jaettu vanhat klassikot sekä omatekoiset projektit, ja yläkerrassa yritykset esittelivät nykyaikaista tarjontaansa. Uusissakin laitteissa oli jäljitelty vanhojen syntetisaattoreiden ominaisuuksia osittain tai suoraan kloonattu alkuperäinen vuosikymmeniä sitten markkinoilta poistunut soitin. Epäuskoisia varten vierellä oli useista soittimista toki tarjolla verrokkina alkuperäinen malli.

Ryhmä laajenee ja poikii uusia

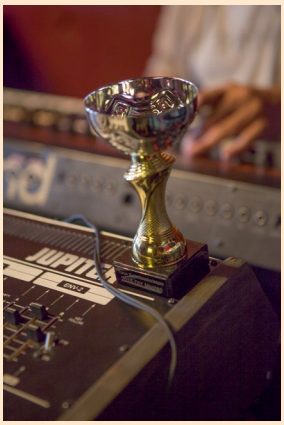
Synamiitin lisäksi Turun syntetisaattoriseuran aktiivisuus on synnyttänyt muutakin hyvää. Kokeellisen Musiikin Klubi järjestää keikkoja Turun Kirjakahvilassa yhteistyössä Turun elävä musiikki ry Telmun kanssa, ja musiikkikirjastolla on pidetty luentoja sekä työpajoja. Seuran kautta toisensa löytänyt tee-se-itse-kolvausporukka tapaa noin kerran viikossa. Softasyntetisaattoriseura on taas haarauma, joka keskittyy vain ei-fyysisiin syntetisaattoreihin.

Muutamiin eri paikkoihin (Tampere, Helsinki, Berliini ja jopa Portugali) on syntynyt aluejaostoja, jotka järjestävät pienimuotoisempia tapaamisia ympäri vuoden. Niistä suuri osa on baarimiittejä, joihin jotkut tuovat mukanaan pienempiä soittimia kokeiltavaksi baaripöydällä. Baaritapaamisten lisäksi seuran jäsenet järjestävät toisinaan koulutustilaisuuksia esimerkiksi modulaarisyntetisaattoreista tai musiikin tuottamisesta. Joskus jäsenet vierailevat toistensa studioilla, perustavat yhtyeitä tai muita yhteistyökuvioita. Keikkajärjestelyt kaupunkien välillä ovat selkeästi helpottuneet verkostoitumisen ansiosta, ja ystävyyssuhteita syntyy runsaasti.

Turun syntetisaattoriseuran Facebook-ryhmän viestitulva kasvoi 2014



Toisinaan itse rakennettujen syntetisaattorien kuuluukin näyttää kotikutoisilta kuten tämä Comic Sans -fontilla koristeltu modulaarisyntetisaattori.

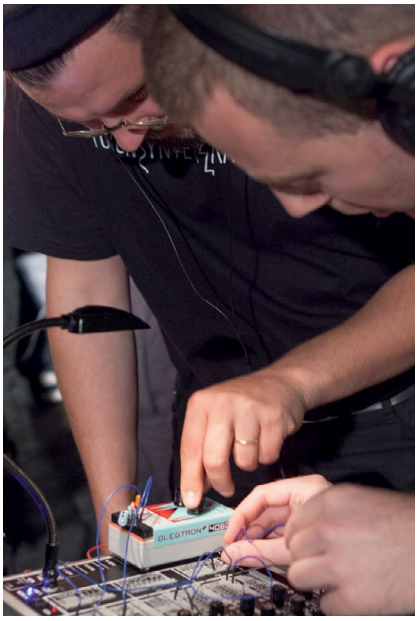


Tällä kerralla jaettiin ensimmäistä kertaa vuoden DIY-projekti -pokaali, jonka voitti Jani Kemppaisen kunnianhimoinen Roland Jupiter 8 -klooni. Jo projektin alkuvaiheessa edistystä seuranneet harrastelijat nostivat hattua tai pitivät rakentajaa hulluna ryhtyessään niin mielettömään urakkaan. Kustannussyistä aivan täydellistä identtisyyttä ei tavoiteltu, vaan äänimäärä puolitettiin neljään samanaikaiseen ääneen. Valmistajan nimi jätettiin tyhjäksi. Muilta osin ulkonäössä pyrittiin mahdollisimman identtiseen sekä toiminnallisuuteen että ulkoasuun, kytkimiä ja rungon merkintöjä myöten. Komponenteissa suosittiin lähituotantoa, mikäli se ei vaikuttanut ääneen. Aivan valmiiksi projekti ei ehtinyt ennen tapahtumaa. Vielä viimeisenä yönä rakentelu-urakka venyi pikkutunneille, kun saunonnan vilvoittelutauolla juotellessa sisään tippuneet hikipisarat sekoittivat laitteen toimintaa. Toiminnoista osa jäi vielä toteuttamatta, mutta äänipuolen osalta kotimaista kopiota saattoi vertailla vastapäätä sijoitettuun alkuperäiseen malliin, ja tulos oli kokeilijoiden mukaan hyvin lähellä alkuperäistä.

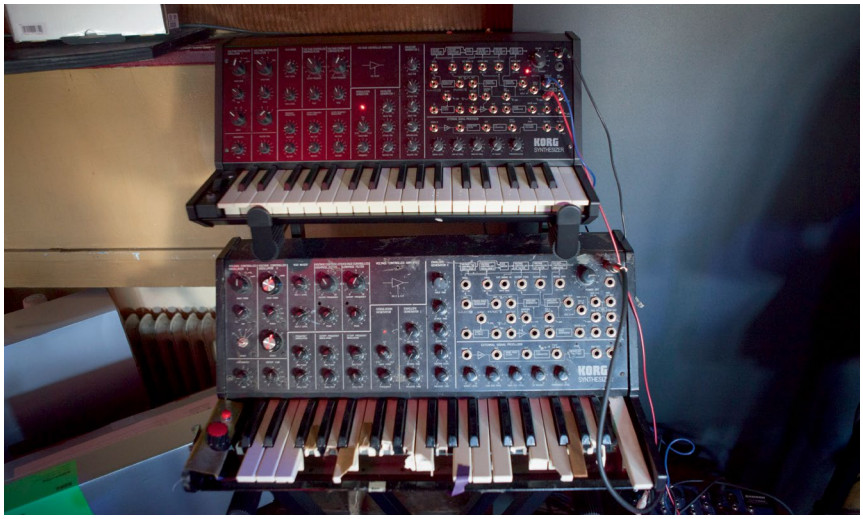
Jani on kunnostautunut aiempinakin vuosina huikeilla omatekoisilla projekteillaan. Maininnan arvoisia ovat ainakin Roland TR-808- ja ARP 2600 -kloonit.

Vastakkain alkuperäinen Roland Jupiter-8 sekä suomalainen halpakopio, joka tuntui vetävän kokeilijoita huomattavasti esikuvaansa enemmän.

Aiempien vuosien kohokohtia on ollut muun muassa laadukkaista ääniprosessointilaitteistaan tunnetun kotimaisen Knif Audion kokonaan putkista valmistettu Knifonium, joka on rakennettu itse koskettimia myöten. Ulkoasussa on piirteitä menneistä vuosikymmenistä sekä vuosisatojen takaisista cembaloista. Ääni on hyvin persoonallinen. Viisinumeroisen hintalapun vuoksi valmistajan läsnäolo tapahtumassa tarjosi suurelle osalle seuran jäsenistä ainoan mahdollisuuden päästä kokeilemaan soitinta. Jonoa laitteen äärellä riitti läpi tapahtuman. Soitin on käytössä muun muassa nimekkäillä ammattilaisilla, kuten elokuvasäveltäjä Hans Zimmerillä.



Mynämäkeläisestä käsityöpajasta ilmestynyt Olegtron 4060 on rakennettu yksinkertaisesti CMOS-logiikkapiiri 4060:n pohjalle. Se tarjoaa käyttäjälle kahden säätimen lisäksi vain kytkennät eri nastoihin ja oskillaattorijakajiin. Koekytkentälevyyn voikin uskaliaasti latoa sekalaisia elektroniikan komponentteja tai muita instrumentteja. Olegtron toimii laajalla jännitealueella paristosta muuntajaan, jopa 18 volttiin.



Kokeiltavana Korgin uusi miniversio MS-20:sta sekä elämää nähnyt, Jimi Tenorilta lainaan saatu alkuperäinen. Artisti väittää pystyvänsä soittamaan syntetisaattorilla sooloja huolimatta siitä, että koko laitos tuntui painuvan kasaan jo kosketeltaessa.

Tapahtumaillan päättää erillinen konsertti. Kuvassa Velcro Fastener, joka sai kesken esityksensä vastailla yleisön puristien kuittailuun liian moderneista laitteista.

niin suureksi, että oli tarve perustaa erillinen ryhmä, jossa saa keskustella myös aiheen ohi (Synaseura B). Myöhemmin perustettiin myös haararyhmä Softasyntetisaattoriseura.

Facebook-ryhmän ohella toimii myös web-foorumi, vaikkakin sen päivitys on ollut viime aikoina hiljaista. Foorumilla jäsenten jakama arvokas tieto ei huku viestitulvaan niin helposti kuin Facebookissa. Tiedon helmien arkistoimisesta on ollut puhetta, ja myös seminaarien ja työpajojen videotaltiointi on ollut harkinnassa. Budjetti ja työmäärä ovat kuitenkin muodostuneet suunnitelmien esteeksi.

Tulevaisuus

Seuran ja tapahtumien kasvamisen suurimpana tavoitteena on yhteistyötoiminnan kehittäminen. Kulisseissa tapahtuu parhillaan kaupunkien ja maiden välistä ideavaihtoa, kustannustoimintaa ja tapahtumajärjestelyä.

Yksi vahvimmin kehittyvistä osa-alueista on softasynaseura, jonka fyysiset tapaamiset on tarkoitus saada käyntiin alkavana talvena. Alaseuran jäsenissä on loppukäyttäjien lisäksi ohjelmistokehittäjiä, frontend-käyttöliittymäsuunnittelijoita, alfa- ja betatestaajia, rautaosaajia ja tehdasäänisuunnittelijoita. Näiden eri osaajien yhteistyötoiminnan seurauksena syntyy parhaimmillaan uusia tapahtumia, yhtyeitä tai jopa startup-hankkeita.

Äänisynteesi
Kohinasta kanttiaaltoon

Tietokoneet pystyvät tuottamaan ääntä monenlaisilla tekniikoilla. Millaisia mahdollisuuksia tähän ylipäätään on olemassa, ja saisiko softasynan ohjelmoitua itse?

Teksti: Ville-Matias Heikkilä Kuvat: Sakari Leppä, Chris Helenius, Mitol Meerna, Yrjö Fager

Lyhyesti määriteltynä ääni on korvin kuultavia värähtelyjä ilmassa tai muussa väliaineessa. Ääntä tuotetaan monin tavoin: kielisoittimet perustuvat pingotettujen lankojen värähtelyyn, ihmisääni ja puhallinsoittimet ääniaaltojen kimpoilemiseen seinämistä ja kaiuttimet sähkömagneetin avulla liikuteltavaan kalvoon.

Tietokoneetkin ovat pitäneet ääntä alusta asti. Aluksi ääni oli enimmäkseen releitten ja reikäkorttien mekaanista naksuntaa, mutta jo 50-luvun alkupuolella tietokoneisiin alettiin kytkeä kaiuttimia. Tyypillinen ratkaisu oli kytkeä kaiutin akkurekisterin ylimpään bittiin. Tästä oli hyötyä paitsi tavallisessa operaattorintyössä – ääntä seuraamalla pystyi päättelemään, edistyikö laskenta vai oliko ohjelma juuttunut silmukkaan – mutta se mahdollisti myös musiikin soittamisen. Vakioyhteenlaskua toistamalla ylin bitti kääntyi nollasta ykköseksi ja takaisin vakioajassa, mikä sai aikaan tasakorkuisen vinkunan. Vakion muuttaminen muutti äänen korkeutta.

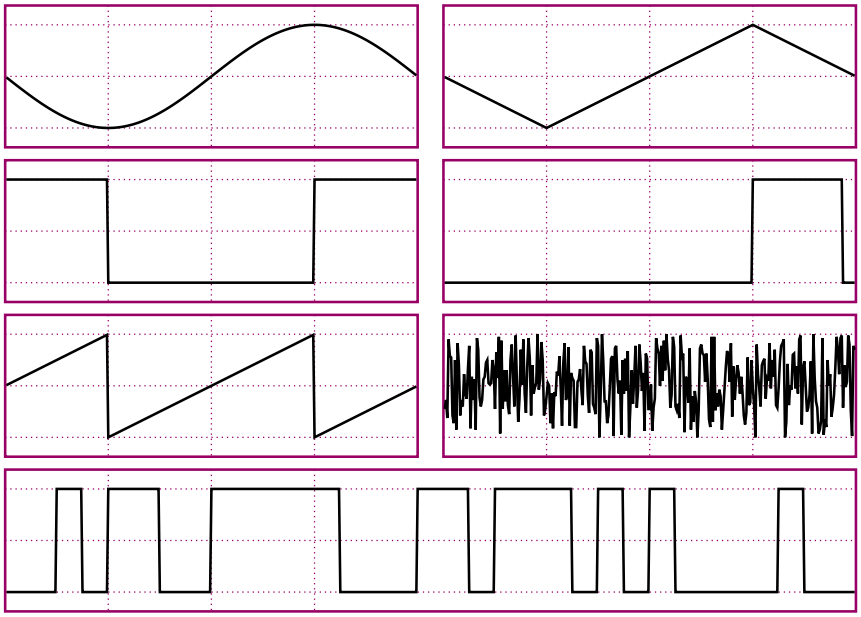
Mikrotietokonemaailman yksinkertaisimmat äänilaitteet – esimerkiksi ZX Spectrumin piipperi – toimivat likimain samalla periaatteella: kaiutinta ohjaa yksi I/O-bitti, jota vaihdellaan ohjelmallisesti. Tämä toki vie paljon suoritinaikaa. PC-piipperi on hieman kehittyneempi: sitä ohjaa oma ajastin, joten vakiokorkuisen piippauksen ylläpito ei vaadi jatkuvaa suoritintyötä. Sen sijaan jos PC-piipperillä halutaan soittaa kahta sävelkorkeutta samaan aikaan, ei ajastimesta ole enää mitään hyötyä.

Sekä alkuperäisen Spectrumin että PC:n äänio minaisuudet ovat varsin minimaaliset verrattuna saman aikakauden koneisiin, joissa on äänentuotantoon erikoistunutta apurautaa. Tyypillisessä 80-luvun alkupuolen äänipiirissä on kahdesta neljään äänikanavaa, joiden äänenkorkeutta, aaltomuotoa ja äänenvoimakkuutta voi säätää. Commodore 64:n SID-piirissä on tarjolla myös ohjattavat suotimet ja mahdollisuus käyttää äänikanavia toistensa moduloimiseen.

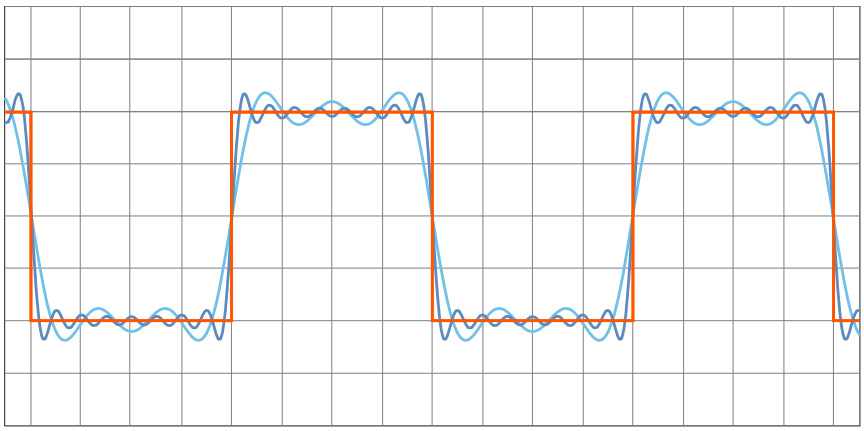
Piipperiä voi parantaa myös lisäämällä sen bittimäärää. PC-maailman yksinkertaisin ”äänikortti” on Covox, joka on käytännössä 8-bittinen D/A-muunnin – laite, joka pystyy tuottamaan 256 mahdollista jännitetasoa peruspiipperin kahden tason sijaan. Covoxin ulostuloa tuhansia kertoja sekunnissa muuttamalla on mahdollista toistaa periaatteessa mitä tahansa ääntä. Kun ääni esitetään peräkkäisinä numeroina, jotka kuvaavat jännitetason vaihtelua, puhutaan *pulssikoodimodulaatiosta* (PCM).

Amigan Paula-piirin äänio minaisuudet perustuvat neljään 8-bittiseen PCM-kanavaan, jotka lukevat soitettavaa ääntä annetulta muistialueelta. Muistialueen lisäksi myös soittonopeutta ja äänenvoimakkuutta voi säätää kanavakohtaisesti.

Nykyisin tietokoneet eivät kovinkaan usein musiikkia soittaessaan tuota sitä reaaliajassa – ne yksinkertaisesti vain purkavat sitä mediatiedostoista, eivätkä ”alkeellisimmat” äänirajapinnat edes tarjoa ohjelmoijalle muuta mahdollisuutta. Jos rajapinnan suunnittelija on kuitenkin ollut tolkuissaan, tarjolla on myös PCM-äänipuskuri, jota ohjelma voi täyttää äänidatalla, joka voi olla hyvinkin raskailla algoritmeilla laskettua.



Yksinkertaisia aaltomuotoja: sini, kolmio, kantti, pulssi 22 %:n suhteella, sahalaita, valkoinen kohina ja eräs Atarin käyttämä LFSR-aaltomuoto.



Kanttiaallon voi rakentaa erillisistä siniaalloista seuraavaan tapaan.

Ääniaallolla on muoto

Sähköpiiriä, joka tuottaa samanlaisena toistuvaa signaalia, kutsutaan *oskillaattoriksi*. Kun tätä signaalia kuunnellaan äänenä, on ääni sitä korkeampi, mitä tiheämpää toistuminen on (taajuus tai aallonpituus) ja sitä voimakkaampi, mitä suurempia signaalin voimakkuuserot ovat (amplitudi). Sointiväri puolestaan riippuu voimakkuuserojen muodostamasta kuviosta eli aaltomuodosta.

Kanttiaalto on digitaalisessa logiikassa yksinkertaisin aaltomuoto. Se syntyy, kun signaali vaihtelee säännöllisin väliajoin jyrkästi kahdesta ääripäästä toiseen – tietokoneen kannalta siis esimerkiksi ykkösen ja nollan välillä. Lähes kaikki digitaaliset äänipiirit pystyvät tuottamaan kanttiaaltoa, ja musiikissa käytettynä se onkin monen korvaan kaikkein ”tietokonemaisin” tai ”kasibittisin” aaltomuoto.

Kun kanttiaallon kahden vaiheen kestot poikkeavat toisistaan, puhutaan *pulssiaallosta*. Kestosuhteen vaihtelulla eli pulssinleveysmodulaatiolla (PWM) aikaansaatavat naukunat ja pörinät ovat erityisen tyypillisiä SID-musiikille. Monissa nykyisissä digitaalilaitteissa käytetään PWM:ää vapaamuotoisen äänisignaalin tuottamiseen: kun korkeataajuuksinen pulssiaalto ohjataan alipäästösuotimen läpi, pulssinleveyserot muuttuvat signaalinvoimakkuuden eroiksi.

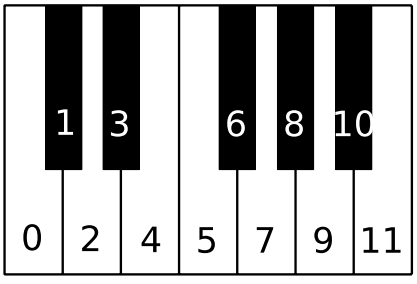
Siniaalto on perustavanlaatuinen aaltomuoto sekä analogisissa oskillaattoreissa että äänen matemaattisessa analyysissä. Kaikenlainen ääni voidaan hajottaa eritaajuisten ja -voimakkuuksisten siniaaltojen summaksi Fourier-muunnokseksi kutsutulla toimituksella. Yksittäinen siniaalto edustaa tässä puhdasta taajuutta – ääntä, joka ei ole hajotettavissa alkeellisempiin harmonisiin osiin.

Kolmioaalto on korvakuulolta samantapainen pehmeä vinkuna kuin siniaaltokin, mutta se on helpompi tuottaa digitaalilogiikalla. Siinä signaalinvoimakkuus muuttuu suoraviivaisesti ääripäästä toiseen siirtyessään, joten aalto on kulmikas.

Sahalaita-aalto eroaa kolmioaallosta siten, että aallon huipulta palataan hyvin jyrkästi suoraan pohjalle. Se kuulostaa paljon läpitunkevammalta kuin kantti- tai kolmioaalto. Se on myös helppo tuottaa ohjelmallisesti yksinkertaisena ympärikierähtävänä laskurina.

Kohina on ääntä, jolla ei ole säännöllisesti toistuvaa jaksoa lainkaan, tai sitten se on niin pitkä, ettei sitä kuulla äänenkorkeutena. Tavanomaisella satunnaislukugeneraattorilla saatava ääni on niin sanottua valkoista kohinaa, joka on tasainen sekoitus kaikkia aallonpituuksia – samaan tapaan kuin valkoinen luonnonvalokin. Jos kohina keskittyy matalammille taajuuksille ja heikkenee taajuuden noustessa, voidaan puhua esimerkiksi ruskeasta, punaisesta tai vaaleanpunaisesta kohinasta.

Klassisissa äänipiireissä kohina tuotetaan useimmiten LFSR:illä eli lineaarisilla takaisinkytkevillä siirtorekistereillä (linear feedback shift register). LFSR:t tuottavat pitkiä ykkösten ja nollien sarjoja, jotka kuulostavat erilaiselta matalalla ja korkealla taajuudella. LFSR-piirit pystyvät tuottamaan myös lyhyempiä sarjoja, joita kuulee erityisen paljon Atarin TIA- ja POKEY-piirien tuottamissa särisevissä äänissä. Myös esimerkiksi SIDin kohinageneraattorin saa tarkalla ajoituskikkailulla huijattua tuottamaan lyhyempää sarjaa.



Länsimainen sävelasteikko puolisävelaskelten määrää kuvaavine järjestyslukuineen.

Äänen laskuoppi

Edellä esitellyt perusaaltomuodot eivät suinkaan ole ainoat mahdolliset – ne vain sattuvat olemaan matemaattisesti yksinkertaisimmasta päästä. Aaltomuodoksi voi vapaasti piirtää millaisen käppyrän hyvänsä, ja näin usein tehdäänkin tracker-pohjaisessa chip-musiikissa. Kaikilla samanlaisina toistuvilla aaltomuodoilla on kuitenkin yksi yhteinen piirre: siniaalloiksi hajotettuna niissä on vain sellaisia aallonpituuksia, jotka menevät tasan niiden kokonaispituuteen.

Jos pisimmän siniaallon taajuus on vaikkapa 400 hertsiä, voi ääneen siis kuulua myös 800, 1200 ja 1600 hertsin ja vastaavasti sitä korkeampia taajuuksia. Pisimmän siniaallon edustamaa ääntä kutsutaan perusäänekseksi, ja se samaistuu äänen sävelkorkeuteen. Sen moninkertoja puolestaan kutsutaan harmonisiksi osaääneksiksi tai osa- tai yläsäveliksi. Luonnollisten soitinten ääni ei ole näin kliinistä: niissä on myös niin sanottuja epäharmonisia osaääneksiä, joiden taajuus ei ole perustaajuuden moninkerta.

Huomattava osa musiikista perustuu siihen, kuinka sävelet soivat peräkkäin ja samanaikaisesti. Jotta eri sävelkorkeudet sointuisivat yhteen, on niiden oltava keskenään sopivissa matemaattisissa suhteissa. Jos sävelkorkeus on kaksi kertaa niin korkea kuin toinen, niiden välistä suhdetta eli intervallia kutsutaan oktaaviksi. Suhdetta 3/2 kutsutaan vastaavasti kvintiksi ja suhteen 4/3 lähelle menevää intervallia kvartiksi. Yksinkertaisilla murtolukusuhteilla eri sävelten äänekset sattuvat lähelle toisiaan, ja erään teorian mukaan yhteensointuvuus eli konsonanssi perustuu juuri tähän.

Länsimaisessa musiikissa oktaavi jaetaan lähes aina kahteentoista sävelkorkeuteen, ja soittimet viritetään useimmiten siten, että puolisävelaskel eli peräkkäisten sävelkorkeuksien välinen suhde on vakio, 2 potenssiin 1/12. Näin saadaan joukko sävelkorkeuksia, joiden väliset suhteet menevät kohtuullisen lähelle murtolukusuhteita. Tietokonemusiikissa tämä niin sanottu tasaviritys on lähes yksinomainen, vaikka kyseessä onkin sinänsä tarpeeton kompromissi. Tietokonetta eivät rajoita soitinten fyysiset puitteet, joten sävelasteikon voisi tarvittaessa laskea uudelleen vaikka joka soinnunvaihdoksella.

Synteesiä on monenlaista

Kun yksinkertaisia aaltomuotoja summataan yhteen, ne soivat samaan aikaan mutta pysyvät muuten sellaisinaan. Tähän perustuvaa synteesitekniikkaa kutsutaan lisääväksi eli *additiiviseksi* synteesiksi. Tietokonemaailmassa sitä edustavat etenkin ”kasibittiset” äänipiirit ja chip-musiikki yleensä.

Tiukimmin ja teoreettisimmin määriteltynä additiivisessa synteesissä rajoitutaan pelkkiin siniaaltoihin, joita summaamalla on mahdollista rakentaa millaisia aaltomuotoja tai ylipäätään ääniä hyvänsä. Tämä toteutuu puhtaimmillaan venäläisinsinööri Jevgeni Murzinin 1950-luvulla valmistamassa ANS-syntetisaattorissa, joka synnyttää lasilevylle piirretyn kuvion mukaisen äänen 144 eritaajuisesta siniaallosta. Nykyaikana näppärin tapa muodostaa ääni tyhjästä loihditun spektrogrammin pohjalta on käänteinen Fourier-muunnos.

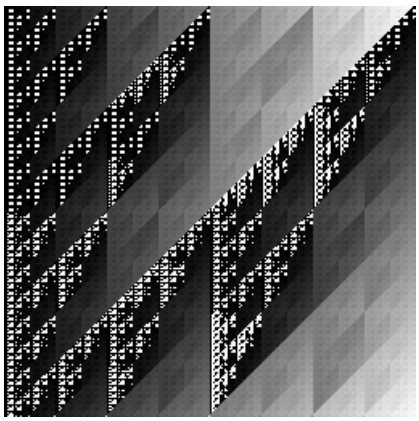
Toisenlainen lisäävä tekniikka on *granulaarisynteesi*, jossa ”ääniatomeja” eli äärimmäisen lyhyitä ääninäytteitä tai vakioaaltomuotoisia pätkiä soitetaan tiiviinä massana eli ”pilvenä” esimerkiksi todennäköisyysjakauman ohjaamana.

Vähentävän eli *subtraktiivisen* synteesin voi mieltää lisäävän synteesin vastakohdaksi: siinä äänestä vaimennetaan eri taajuusalueita suotimien eli filtterien avulla. Termi ei ole kuitenkaan aivan täsmällinen, sillä resonanssisuotimet myös voimistavat joitakin taajuuksia. Suotimet ovat hyvin yleisiä klassikkosyntetisaattoreissa, mutta vanhoissa äänipiireissä ne ovat melko poikkeuksellisia – lähinnä siksi, ettei yksinkertainen digitaalilogiikka oikein taivu niihin. Merkittävin poikkeus on SID analogisuotimineen.

Jos tietokoneesta halutaan ulos mahdollisimman realistista ääntä, on varma ja helppo ratkaisu yksinkertaisesti toistaa tosimaailmasta tallennettuja ääninäytteitä eli sampleja. Amigasta lähtöisin oleva tracker-musiikki perustuu samplejen käyttämiseen soittimina. Samplepohjaiseen musiikkiin saa helposti mukaan kaikenlaisia ääniä, mutta ongelmana on etenkin takavuosikymmeninä ollut niiden runsas tilanvienti – varsinkin, kun aidonkuuloisen soittimen toteuttamiseksi ei yleensä riitä pelkkä yhden samplen toistaminen eri nopeuksilla ja äänenvoimakkuuksilla.

```
main()
{
  int t=0;
  for(;;t++) putchar(t);
}
```

Listaus 1. Sahalaita-aaltoa standardiulostuloon kirjoittava C-ohjelma.



Bytebeat-musiikki usein sekä kuulostaa että näyttää mielenkiintoiselta. Tässä lausekkeen (t*((3+(1^t>>10&5))*(5+(3&t>>14))))>>(t>>8&3) tuottamaa grafiikkaa.

Synteesitekniikoita on yhdistetty monin eri tavoin, jotta saataisiin niiden hyvät puolet käyttöön samalla kertaa. Eräs tällainen yhdistelmätekniikka on Rolandin syntetisaattoreissa ja PC-lisälaitteissa käytetty lineaariaritmeettinen synteesi, joka yhdistää samplattuja elementtejä vähentävään synteesiin.

Aivan toisenlainen lähtökohta synteesille on äänen fysikaalisen syntymekanismin mallintaminen taajuusrakenteen sijaan. Erilaisia mallinnustapoja on monenlaisia, mutta ehkä eniten käytetty on *digital waveguide*, joka sopii hyvin esimerkiksi kieli- ja puhallinsoittimien jäljittelyyn. Mallit kuvaavat ääniaallon etenemistä yksiulotteisissa johtimissa, joiden eri kohtia on kytketty toisiinsa suotimilla.

Äänipuskuria täyttämään

Äänisynteesin perusteisiin saa ehkä parhaan tuntuman, kun toteuttaa itse oman ohjelmallisen syntetisaattorin eli softasynan.

Softasynan ohjelmoijalla on yleensä aluksi vastassa rajapinta, joka ottaa vastaan lukuarvoja – täytettä PCM-puskuriin. Unix-maailmassa musiikkiohjelma saattaa yksinkertaisimmillaan vain kirjoittaa tavuja standardiulostuloon, joka täytyy edelleenohjata ääniajurin edustajalle (/dev/dsp, aplay, pacat). Toiset rajapinnat,

```
char nuotit[16*2] =
{
  26, 14,
  28, -1,
  29,  2,
  33, -1,
  38, 14,
  -1, -1,
  33,  2,
  -1, -1,
  29,  5,
  -1, -1,
  -1, 17,
  31, -1,
  28,  4,
  -1, -1,
  24, 16,
  -1, -1
};

struct
{
  float vaihe,taaj,vol;
} kan[2];

int seuraava_nayte()
{
  if((aika%6000)==0)
  {
    int rivi= aika/6000)%16;
    for(int i=0;i<2;i++)
    {
      int n=nuotit[rivi*2+i];
      if(n<0)
        kan[i].vol/=2;
      else
      {
        kan[i].taaj =
          pow(2.0,n/12.0)*100/48000.0;
        kan[i].vol=0.5;
      }
    }
  }
  aika++;

  float ulos=0;
  for(int i=0;i<2;i++)
  {
    kan[i].vaihe =
      fmod(kan[i].vaihe+kan[i].taaj,1.0);
    ulos+=(kan[i].vaihe-.5)*kan[i].vol;
  }

  if(ulos<-1) ulos=-1;
  if(ulos>1) ulos=1;
  return (int)(ulos*32767);
}
```

Listaus 2. Yksinkertaisen ohjelmallisen syntetisaattorin pohja, joka tuntee vain kanttiaallon.

esimerkiksi SDL, kutsuvat ohjelmoijan määrittelemää vastakutsufunktiota (callback): ”tässä sinulle osoitin puskurin alkuun ja sen pituus, täytä se haluamallasi äänellä”.

Ehkä yksinkertaisin mahdollinen ohjelma, joka tuottaa jonkinlaista ääntä, on ikuinen silmukka, joka toistaa kahden tavun mittaista sarjaa. Kun ohjelman ulostulo ohjataan 8-bittistä monoääntä vastaanottavalle ääniajurille, tuloksena on kimeä kanttivinkuna, jonka taajuus on puolet äänilaitteen näytteenottotaajuudesta. Toinen hyvin yksinkertainen ohjelma on ikuinen silmukka, joka tulostaa silmukkalaskurin

```
INSTRUMENT NUM. 05  Snare
Attack/Decay    04  Vibrato Param   03
Sustain/Release E9  Vibrato Delay   00
Wavetable Pos   0A  HR/Gate Timer   02
Pulsetable Pos  09  1stFrame Wave   09
Filtertable Pos 06

WAVE TBL  PULSETBL  FILT.TBL  SPEEDTBL
06:FF 00  06:30 20  06:90 82  06:00 80
07:81 D8  07:30 E0  07:00 FF  07:00 20
08:41 00  08:FF 06  08:40 F8  08:02 60
09:FF 00  09:88 00  09:FF 00  09:02 00
0A:81 D8  0A:FF 00  0A:90 82  0A:00 18
0B:41 A8  0B:88 00  0B:00 FF  0B:00 10
0C:41 AC  0C:FF 05  0C:FF 01  0C:00 00
0D:80 D4  0D:82 00  0D:90 00  0D:00 00
0E:FF 00  0E:10 40  0E:FF 00  0E:00 00
```

Edistyneetkin SID-musiikkiohjelmat ovat varsin heksatäyteisiä. Tässä on määritelty virvelirumpu, johon kuuluu yksi kehys kohinaa (81), kaksi pulssia (41) ja loput äänestä ovat vaimenevaa kohinaa (80). Heksakoodi FF 00 on listan loppumerkki.

arvon alabittejä – eli lukusarjaa 0..255 loputtomiin. Äänenä tämä on sahalaita-aalto, jonka taajuus on 256-osa näytteenottotaajuudesta – esimerkiksi Unix-äänirajapintojen 8000 hertsin oletusarvolla 31,25 Hz. Tämän C-kielinen toteutus on listauksessa 1.

Kun esimerkkiohjelman putchar()-funktion parametria `t` monimutkaistetaan, saadaan erimuotoisia, -korkuisia ja -voimakkuuksisia aaltomuotoja ja yllättävän pienellä koodilla myös musiikin kuuloista ääntä. Äärimmäisimmästä päästä lyhyydessä on kaava `t&t>>8`, joka muokkaa kanttiaaltoa kytkemällä sen ulostulobittejä päälle ja pois 256 kertaa hitaammin etenevän laskurin mukaan. Jokainen ulostulobitti voidaan ajatella omaksi kanttiaallokseen, jonka ihmiskorva helposti kuulee erillisenä äänenä, jos se alkaa soida yhtäkkisesti. Jos sekaan lisätään kertoimia, jotka eivät ole kahden potensseja – kuten vaikkapa kaavassa `(t*5&t>>7)|(t*3&t>>10)` – ilmestyy musiikin rakenteeseen myös muita intervalleja kuin oktaaveja.

Näiden niin sanottujen bytebeat-ohjelmien toimintaa voi kokeilla myös www-sivuilla – esimerkiksi Skrollin sivulla osoitteessa skrolli.fi/bytebeat/. Bytebeateja etsiskeltiin ahkerasti vuonna 2011, jolloin mielenkiintoisia kaavoja löytyi toisinaan jopa umpimähkäisillä kokeiluilla. Tekniikka ei kuitenkaan päätynyt laajempaan käyttöön esimerkiksi demoskenellä, jossa se olisi saattanut tarjota mahdollisuuden käyttää musiikkia jo parinsadan tavun mittaisissa tuotoksissa. Suosion hiipuminen johtui ehkä tekniikan vaikeasta hallittavuudesta ja äänimaailman yleisestä rujoudesta.

Näin se tehdään perinteisesti

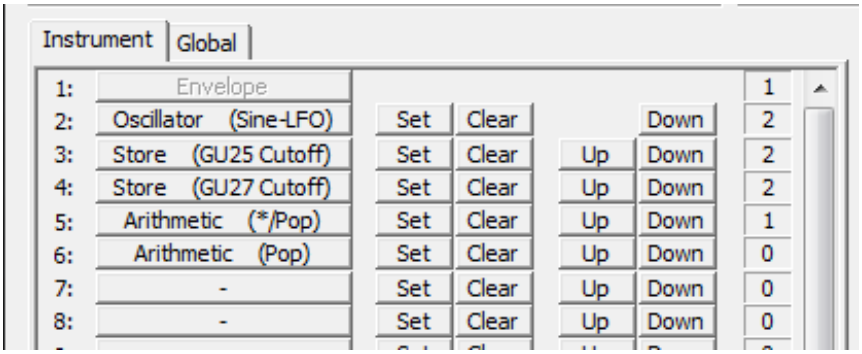
Tavalliset softasynat eroavat huomattavasti bytebeat-ohjelmista. Ne jakautuvat selkeästi erillisiin osiin – esimerkiksi matalan tason synteesi on erillään ”nuotinlukijasta”. Melodiat ja rytmit eivät synny oudoista bitinsekoituskaavoista ja laskennan epätarkkuuksista, vaan ne on tallennettu helposti hallittaviin tietorakenteisiin.

”Vakavasti otettavat” sävellysohjelmat jäljittelevät nuottidatan tallennustavassa nuottikirjoitusta: kullekin nuotille tai tauolle on määrätty erillinen kesto. Perinteisessä demo-, tracker- ja chip-musiikissa käytetään sen sijaan taulukoita, joita kelataan läpi vakionopeudella, ja nuotin kesto riippuu siitä, montako tyhjää riviä kanavalle jää ennen seuraavaa nuottia. Tämä tallennustapa on usein tietokoneen kannalta edullisempi, mutta sillä on vaikea kuvata monimutkaisempia rytmirakenteita. 4/4-tahtista tusinapoppia väsätessä tämä ei kuitenkaan ole mikään ongelma.

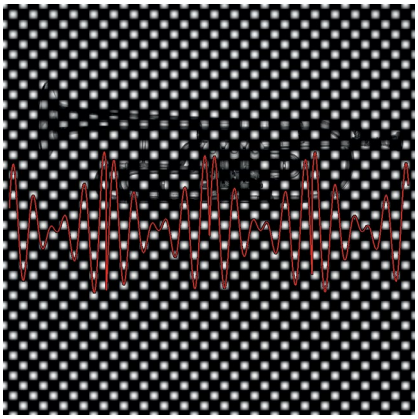
Listaus 2 kuvaa yksinkertaista softasynan runkoa, johon kuuluu trackertyylisenä taulukkona tallennettu kaksiääninen nuottikuvio. Taulukossa edetään yhden rivin verran eteenpäin aina 6000 näytteen välein – eli 48000 hertsin toistotaajuutta käytettäessä 8 kertaa sekunnissa. Etenemisen yhteydessä kanavalle asetetaan uusi taajuus ja äänenvoimakkuus. Taajuus lasketaan potenssiinkorotuksella länsimaisen 12 puolisävelaskelen tasavirityksen mukaan: nuottitaulukon luku 0 tarkoittaa 100 hertsin perustaajuutta, luku 1 sitä yhden puolisävelaskeleen verran korotettuna ja niin edelleen. Luku -1 on taukomerkki, jonka kohdalla kanavan volyymi puolitetaan.

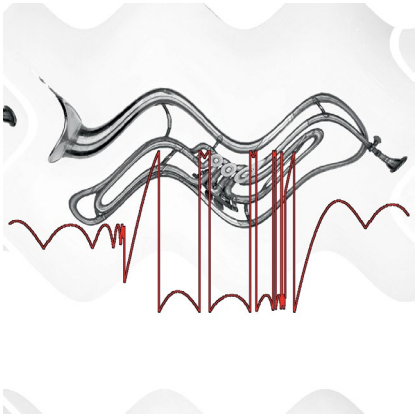
Esimerkkiohjelma tuntee vain yhden aaltomuodon – sahalaita-aallon, joka saadaan suoraan kanavan vaihelaskurin arvosta. Siitä tosin vähennetään puoli aallonkorkeutta, jotta se asettuisi symmetrisesti nollan molemmin puolin. Vaihelaskuri pidetään nollan ja yhden välillä jakojäännösoperaatiolla (`fmod`). Aaltomuotoa voi vaihtaa muuttamalla `ulos+=` -alkuisen rivin sisältöä – esimerkiksi itseisarvon (`fabs`) avulla sen saa muutettua puolta hiljaisemmaksi kolmioaalloksi.

Klassisessa kotitietotekniikassa käytetään itsenäisesti ääntä syntetisoivia äänipiirejä, joiden toimintaa ohjataan tasaisin väliajoin kutsuttavassa soitto-

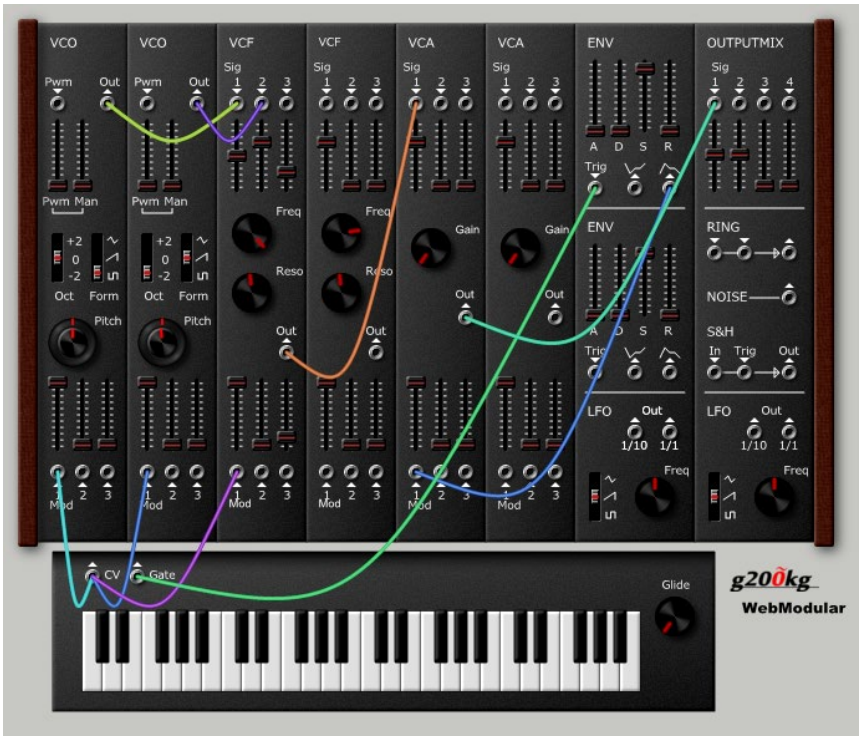


4klangin soittimet perustuvat alkeisfunktioihin, jotka ketjutetaan pinomuistin avulla. Oikeanpuoleisin sarake osoittaa, paljonko pinossa on arvoja operaation jäljiltä.





Näin sahalaita-aalto amplitudi- ja taajuusmoduloituu siniaallon ohjaamana. Trumpetin kuvat esittävät, mitä samaiset algoritmit tekevät grafiikalle.



Jotkin palikanyhdistelyyn perustuvat softasynat muistuttavat käyttöliittymältäänkin modulaarisyntetisaattoreita. Kuvassa WebModular.

rutiinissa. Soittorutiini päivittää äänipiirin rekistereitä eli tekee samoja asioita, mitä esimerkkiohjelma tekee kan-taulukolle seuraavaan nuottiin siirtyessään.

Soittorutiinia kutsutaan esimerkiksi SID-musiikissa yleensä 50 kertaa sekunnissa eli joka näytönvirkistyksellä, eikä se suinkaan siirry seuraavaan nuottiin joka kerralla. Sen sijaan se voi vaihdella äänen parametreja ja saada siten aikaan siihen lyhyemmän aikavälin dynamiikka. Esimerkiksi tyypillisessä SID-rummussa vaihdellaan eritaajuuksisten kohina- ja pulssiaaltojen välillä. Sointuja voi puolestaan soittaa yhdellä kanavalla toistamalla sen nuotteja silmukassa peräkkäin (arpeggio).

Instrumenttikohtaisten komentosarjojen läpikäymisen lisäksi soittorutiini voi myös suorittaa efektejä, esimerkiksi sävelkorkeuden vähittäistä liu'utusta nuotista toiseen. Tämäntyyppisiä efektejä käytetään erityisen paljon samplepohjaisessa tracker-musiikissa.

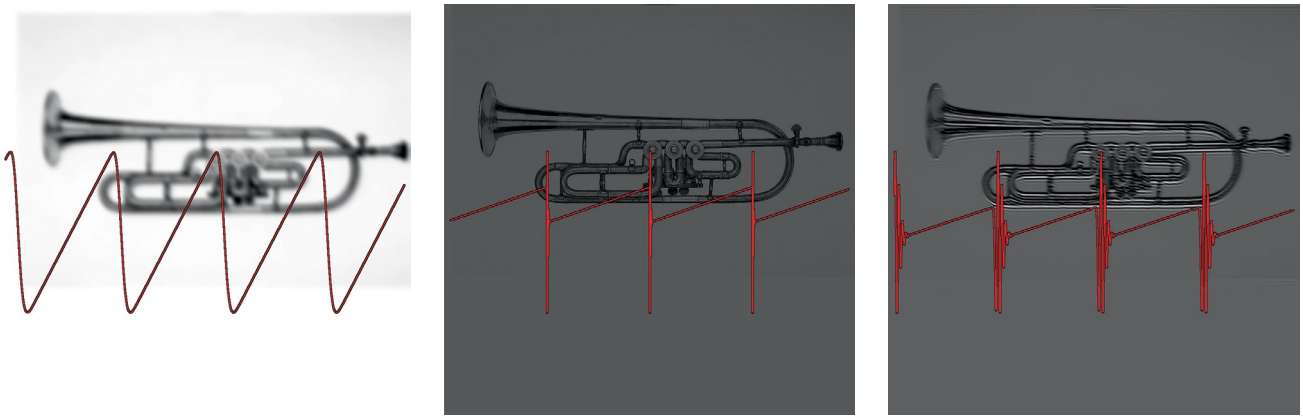
Lisää dynamiikkaa

Softasynassa ei tietenkään tarvitse rajoittua yksinkertaisiin aaltomuotoihin tai chip- ja tracker-musiikin perinteisiin. Varsinainen hupi oikeastaan alkaakin vasta alkaa siitä, kun syntee-sialgoritmia pääsee rikastamaan.

Yksinkertaisella aaltomuoto-oskillaattorilla on yleensä kaksi parametria – taajuus ja äänenvoimakkuus. Jos nämä pysyvät vakioina koko nuotin ajan, on tuloksena varsin konemainen piippaus. Sen sijaan jos näitä parametreja muutellaan eli *moduloidaan* esimerkiksi toisen oskillaattorin ohjaamana, saadaan ääneen jo huomattavasti enemmän luonnollisuutta.

Kun äänenvoimakkuutta säädetään niin sanotulla verhokäyrällä, sen saa muistuttamaan esimerkiksi luonnollisen soittimen äänenvoimakkuuden vaihtelua. Verhokäyrät kuvataan tyypillisesti neljällä parametrilla: *attack* (nopeus, jolla hiljaisuudesta siirrytään maksimivoimakkuuteen), *decay* (nopeus, jolla ääni vaimenee perusvoimakkuuteen), *sustain* (perusvoimakkuus) ja *release* (nopeus, jolla ääni vaimenee takaisin nollaan). Verhokäyrä pysyy perusvoimakkuudella yleensä niin kauan, kuin ”kosketin on alhaalla” (key on), minkä jälkeen se siirtyy release-vaiheeseen. Käyrä voi toki olla ADSR-mallia huomattavasti monimutkaisempikin, kuten esimerkiksi 90-luvun jälkipuoliskon PC-träkkereissä.

Kun taajuutta tai äänenvoimakkuutta moduloidaan jatkuvalla aaltomuodolla, esimerkiksi siniaallolla, saadaan vibrato tai tremolo. Mikäli modulaattorin taajuus on vähintään samaa suuruusluokkaa itse kantoaallon kanssa, saa-



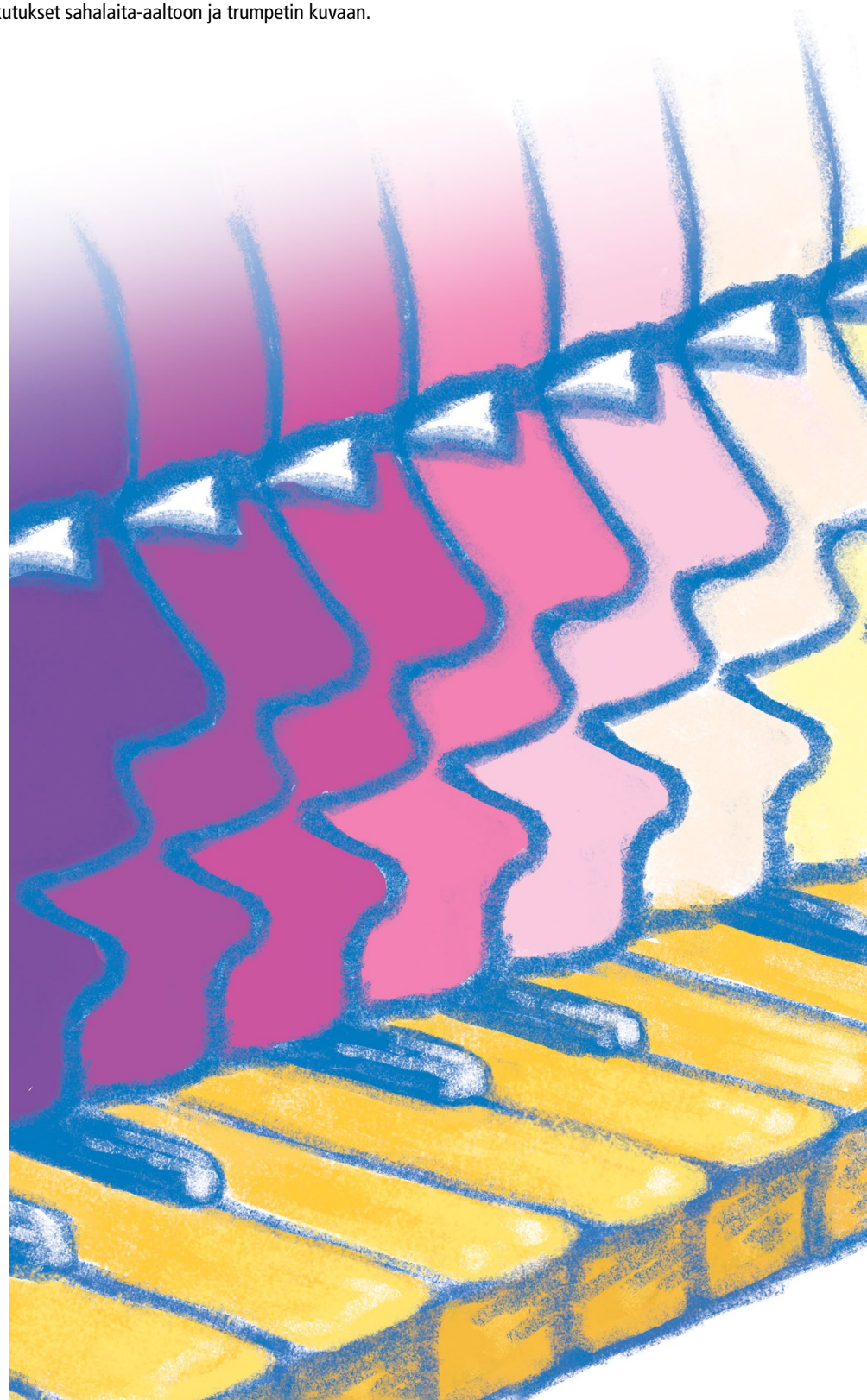
Alipäästö-, ylipäästö- ja resonanssisuotimen vaikutukset sahalaita-aaltoon ja trumpetin kuvaan.

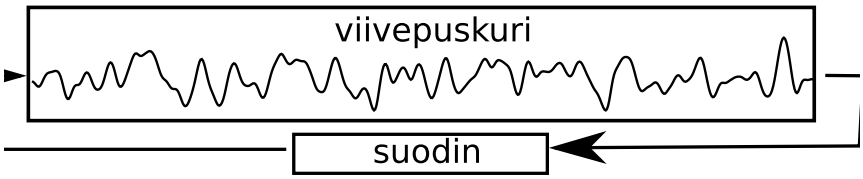
daan taajuus- tai amplitudimodulaatio (FM tai AM). Amplitudimodulaatiota kutsutaan usein myös rengasmodulaatioksi sen toteuttavan analogisen piirin muodon mukaan. Etenkin taajuusmodulaatiossa ääneen syntyy paljon epäharmonisia ääneksiä, joita ei esiinny kummassakaan alkuperäisessä aaltomuodossa. FM-synteesiä on tietokonemaailmassa käytetty etenkin PC:n Adlib- ja SoundBlaster-äänikorttien OPL-piireissä, minkä vuoksi sillä on ehkä tarpeettomankin huono maine tietokoneharrastajien keskuudessa.

Toki myös monenlaisia muita parametreja voidaan moduloida verhokäyrin ja oskillaattorein. Muutamana esimerkkinä vaikka pulssiaaltomuodon leveyssuhde, suotimen kynnystaajuus ja kahden eri aaltomuodon sekoitussuhde. Jos äänigeneraattorina on jokin muu kuin tavanomainen aaltomuotooskillaattori, voi sillä olla aivan toisenlaiset säädettävät parametrit. Toki modulaation lähteenä voi käyttää myös muiden modulaatioiden tuloksia – yhdistelymahdollisuudet ovat liki rajattomat.

Erilaisten synteesifunktioiden yhdistelyä voi kokeilla hyvin paitsi perinteisillä ohjelmointikielillä, myös *Supercolliderin* kaltaisilla äänisynteesiin tarkoitetuilla ohjelmointiympäristöillä. Monet ohjelmat antavat yhdistellä palikoita graafisesti joko abstraktissa kaavionäkymässä tai aitoa modulaarisyntetisaattoria muistuttavassa käyttöliittymässä.

Palikkaverkostoilla saa luotua varsin monipuolisia ääniä siihen nähden, kuinka pienellä datamäärällä sellaisen pystyy kuvaamaan. Tämä on tehnyt siitä houkuttelevan lähestymistavan esimerkiksi pienikokoisten demojen musiikkirutiineihin. Esimerkiksi





Karplus-Strong-algoritmin toimintaperiaate.

Windows-pohjaisiin 4k-demoihin tarkoitetun *4klang*-softasynan soittimet perustuvat yksinkertaisella pinopohjaisella kielellä kuvattuihin rakenteisiin.

Suodatettua ääntä

Monesti eri synteesitekniikoiden jäljiltä ääneen jää esimerkiksi korkeita vinkunoita ja muita ei-toivottuja taajuuksia. Tällöin äänen taajuuskoostumusta on hyvä tasoittaa suotimien avulla, vaikkei suotimilla muuten haluaisikaan kikkailla. Toki suotimet ovat hyödyllisiä myös itse synteesissä, sillä monesti halutun äänen taajuuskoostumuksen halutaan esimerkiksi muuttuvan tasaisesti ajan mittaan.

Alipäästösuodin hiljentää korkeampia taajuuksia mutta päästää alemmat läpi. Ohjelmallisesti sen voi toteuttaa sekoittamalla näytteeseen jonkin verran edellisen suodatuksen tulosta – tai keveimmillään vaikkapa vain laskemalla niiden keskiarvon. Kuvankäsittelyohjelmissa sitä vastaa sumennussuodin.

Ylipäästösuodin toimii juuri päinvastoin kuin alipäästösuodin, eli jättää jäljelle sen osuuden, jonka alipäästösuodin poistaisi. Se toteutetaan samalla periaatteella, mutta sekoituskerroin asetetaan negatiiviseksi. Ylipäästösuodinta vastaa kuvankäsittelyssä terävöityssuodin.

Resonanssisuodin on synteesin kannalta erityisen mielenkiintoinen, sillä se voimistaa äänestä tiettyjä taajuusalueita. Näin sen avulla pystytään saamaan aikaan niin sanottuja formantteja eli tietylle taajuusalueelle keskittyviä voimistumia. Digitaalisesti toteutettuna resonanssisuodin on hieman aiemmin-mainittuja monimutkaisempi – se laskee painotetun keskiarvon näytteestä ja kahdesta edellisestä suodatustuloksesta.

Puhutun kielen äänne-erot perustuvat paljolti formanttitaajuuksien eroihin. Niinpä varsin suosittu tapa puhesynteesin toteuttamiseen onkin suotimiin perustuva *formanttisynteesi*. ”Kurkusta” tulee tällaisessa mallissa joko kohinaa tai pehmennetyn sahalaita-aallon kaltaista korinaa riippuen siitä, lausutaanko soinnitonta vai soinnillista äännettä, ja tätä ”kurkkuaaltoa” muovataan resonanssisuotimien sarjalla. Suomen kielen kaikkien vokaalien tunnistettavaan tuottamiseen riittää kolme suodinta, mutta täydellisempi ääntöväylän malli vaatii vielä muutamia lisää.

Kimpoilevat ääniaallot

Jos ääneen halutaan kaikua, onnistuu se luontevimmin viivepuskurilla, josta ääneen saadaan mukaan sen omaa historiaa vaikkapa sekunnin takaa. Erilaisia tilavaikutelmia saa paitsi vaihtamalla kaiun kestoa ja voimakkuutta, myös suodattamalla kaikuja ja asettamalla niitä useita päällekkäin. Tilavaikutelmaa luonnollisestikin vahvistaa, jos käytössä on useampi kaiutin, joille äänen kimpoilut lasketaan erikseen. *OpenAL* on positionaalisen äänen tuottamiseen erikoistunut rajapinta, jota on käytetty etenkin peleissä.

Kun kaikuvien tilojen sijaan halutaan mallintaa soittimien akustiikkaa, voidaan siihen käyttää lyhyempiä viivepuskureita, joiden pituus on haluttujen äänten aallonpituuksien luokkaa.

Ehkä yksinkertaisimman mahdollisen fysikaalisen soitinmallin toteuttaa *Karplus-Strong*-algoritmi. Siinä puskuri täytetään aluksi kohinalla, minkä jälkeen sen sisältöä soitetaan toistuvassa silmukassa samalla alipäästösuodattaen sitä. Tämä aikaansaa yllättävän aidonkuuloisen kitaraäänen. Puskuri vastaa tässä mallissa kieltä, jota pitkin aalto etenee törmätäkseen lopulta kiinnityskohtaansa, josta se hieman suodattuneena kimpoaa vastakkaiseen suuntaan. Koska eteneminen on samanlaista molempiin suuntiin, ei molempia suuntia ole välttämätöntä mallintaa erikseen.

Äänen korkeutta voi Karplus-Strong-mallissa muuttaa vaihtamalla puskurin pituutta, ja suodatusfunktion vaihtelu vaikuttaa muun muassa äänen vaimenemisnopeuteen. Ääneen säröä tuottamalla ja sitä takaisin kielten värähtelyksi johtamalla saa aikaiseksi vaikkapa sähkökitaran. Negatiiviset suodatinparametrit muuttavat äänen metalliseksi kilahteluksi.

Karplus-Strongin yleistettyjä versioita kutsutaan *digital waveguide* -synteesiksi: siinä puskurin varrella voi olla useampia erilaisia suotimia ja linkkejä, ja puskureita voi olla useampiakin. Kielisoitinten ohella ne sopivat hyvin etenkin puhallinten mallinnukseen. Waveguide-mallit voi lisäksi yleistää kahteen tai useampaan ulottuvuuteen, jolloin voidaan simuloida esimerkiksi rumpukalvon värähtelyä.

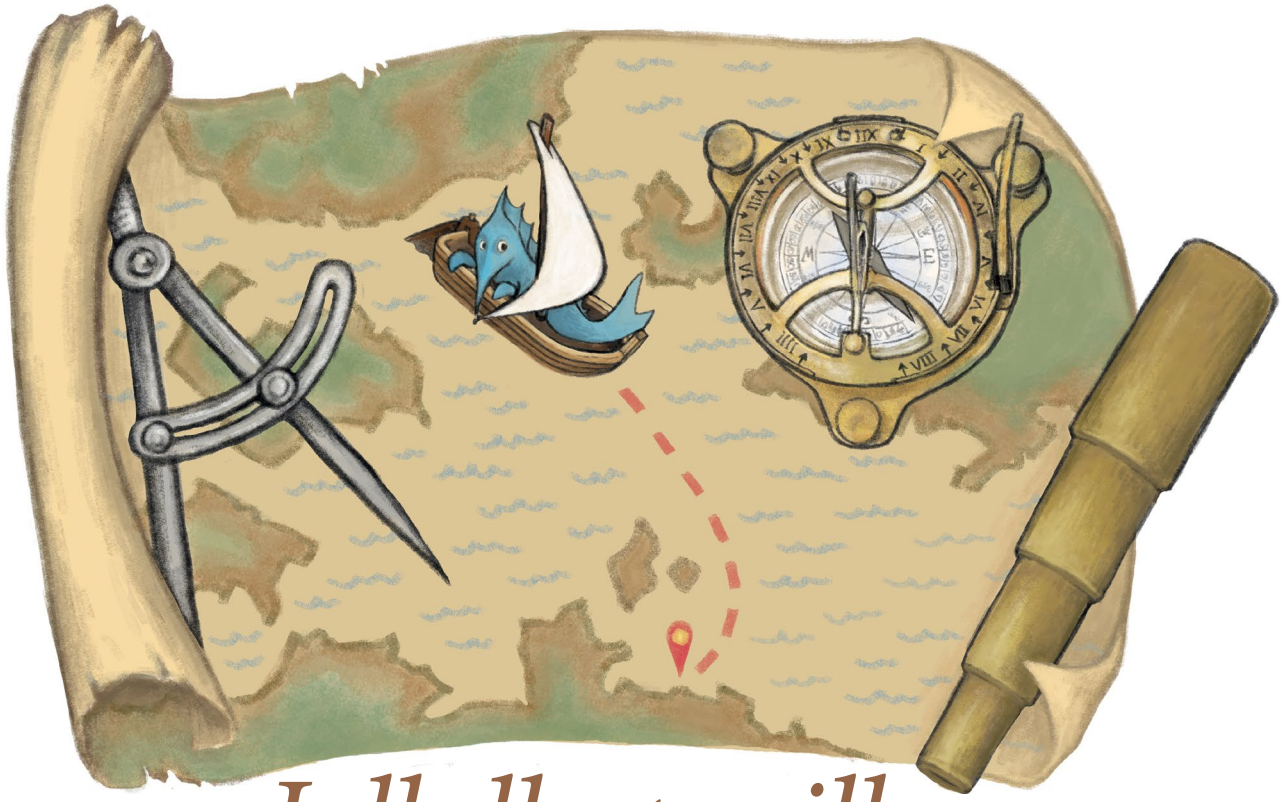
Uusia ulottuvuuksia

Nykyisin eivät laskentaresurssit juuri rajoita sitä, kuinka mielikuvituksellisia synteesihökötyksiä tietokoneella voi rakentaa – olivat ne sitten fysikaalisia malleja, suodin- ja oskillaattoriverkostoja tai vaikkapa niiden yhdistelmiä. Oleellisempi kysymys onkin se, kuinka löytää uusia ja mielenkiintoisia mahdollisuuksia tästä runsaudesta.

Neuroverkkoja on kokeiltu onnistuneesti monilla musiikin- ja äänentuoton osa-alueilla. Rekurrentteja verkkoja voi käyttää jo itsessään syntetisaattoreina: neuronien painokertoimet ovat ikäänkuin potikoita, jotka opetusalgoritmi pyörittelee esimerkiksi vastaamaan annettua malliääntä. Verkko voi toki säätää myös valmiin syntetisaattorin parametreja ja vaikkapa oppia soittamaan ihmiselle aivan liian monimutkaista mielikuvitussoitinta luonnollisen kuuloisesti.

Äänisynteesin seuraavan suuren mullistuksen ei tarvitse välttämättä vaatia tuhottomasti laskentatehoa. On nimittäin takuuvarmasti olemassa myös sellaisia kevyitä synteesimalleja, joita kukaan ei ole vielä tullut ajatelleeksi. Bytebeat-kokeilut paljastivat muutama vuosi sitten, että tietojenkäsittelyn perusoperaatioitakin voi vielä käyttää ennenkokeilemattomilla tavoilla äänisynteesissä, vaikka niiden käyttöön riittävää rautaa on ollut olemassa vuosikymmenten ajan.

Tämän artikkelin tarkoituksena oli antaa jonkinlainen pintaraapaisumainen peruskäsitys äänisynteesistä ja siihen liittyvistä ohjelmointitekniikoista. Omiin kokeiluihin sopivia kieliä, rajapintoja ja leikkimisympäristöjä on valtavat määrät, joten ei muuta kuin etsimään itselle sopiva ja askartelemaan!



Jollalla perille

Sailfish-ohjelmointi, osa 3

Skrollin numerossa 2015.2 päästiin Jolla-puhelimella entisajan navigoinnin huipputekniikkaan, kompassin käyttöön. Nyt hypätään tähän päivään ja opitaan käyttämään satelliittipaikannusta.

Teksti ja koodi: Asser Lähdemäki
Kuvat: Mitol Meerna, Asser Lähdemäki, Mikko Torvinen

Täyden tarkkuuden GPS-satelliittipaikannus on ollut tarjolla kuluttajamarkkinoille vuodesta 2000 ja on vallannut navigointimarkkinat kuluneen 15 vuoden aikana. Sittemmin GPS-paikannus tuli älypuhelimiin ja löytyy tietysti myös Jolla-puhelimista.

Tässä artikkelissa käydään läpi pääpiirteittäin, miten Qt-sovelluskehyksen QML-rajapintojen avulla vastaanotetaan paikkatietoa, kuvataan paikkoja ja esitetään karttoja. Lukijalta oletetaan Sailfish-ohjelmointisarjan edellisissä artikkeleissa käsiteltyjen asioiden tuntemista, jotta vältytään turhalta toistolta.

Esimerkkiohjelmana käytetään numeron 2015.2 artikkelissa käytettyä runkoa, johon on lisätty kolme uutta sivua: paikannus-, kartta- ja paikkasivu. Näistä ensimmäisessä näytetään, miten sijaintitietoa pyydetään rajapinnalta. Toisessa käytetään järjestelmän tarjoamia karttapalveluita kartan näyttämiseen ja sille piirtämiseen. Kolmannessa tutkitaan karttapalveluiden tarjoamia lisäpalveluita, joista esimerkissä käsitellään reititystä ja paikkatietoja eli tietoa ravintoloista ja muista toimijoista. Esimerkkiohjelma on ladattavissa osoitteesta www.skrolli.fi/2015.4.

Sailfish tukee kahta karttapalvelua: Open Street Map (OSM) ja Nokian Here. Esimerkeissä on käytetty OSM-karttoja, koska niiden käyttö ei vaadi rajapinta-avainta, jonka avulla palvelu tunnistaa sovelluksen. Tunnistuksen perusteella palveluntarjoaja laskee käyttötilastoja ja valvoo, että rajapinnan käyttö on sopimuksen mukaista. Kerromme kuitenkin keinot myös Here-karttojen ja -palveluiden käyttöön.

Navigointirajapintojen käyttäminen vaatii ajonaikaisia riippuvuuksia. Sailfish-ohjelmaprojektin .yaml-tiedoston Requires-kohtaan täytyy lisätä riippuvuudet seuraaviin paketteihin:

```
qt5-qtdeclarative-import-positioning >= 5.2
qt5-qtdeclarative-import-location >= 5.0
qt5-plugin-geoservices-osm >= 5.2
qt5-plugin-geoservices-nokia >= 5.2
```

Näistä ensimmäinen on sijaintiin ja paikannukseen liittyvä rajapinta, toinen paikkojen ja karttojen esittämistä varten, kolmas OSM-karttapalvelut tarjoava lisäosa ja viimeinen Here-karttojen lisäosa.

Koordinaattijärjestelmä

Navigoinnissa käytetään koordinaattijärjestelmää, joka pohjautuu pallokoordinaatistoon. Koordinaattijärjestelmässä on määritetty pituus- ja leveyspiirit. Tosiasiassa GPS:n koordinaattijärjestelmä WGS84 (World Geodetic System, 1984) ottaa huomioon maapallon elliptisen muodon, mutta sijaintiin liittyvien laskujen suorittaminen pallokoordinaatistossa säteellä 6371 km antaa yleensä riittävän tarkan lopputuloksen.

Pituuspiirit (longitude) ovat pitkittäin samansuuntaisia maan pyörimisakselin kanssa. Ne siis piirretään maapallon navalta toiselle. Leveyspiirit (latitude) ovat kohtisuorassa pituuspiirejä vasten ja kertovat sijainnin poikkeaman päiväntasaajalta.

Kuvassa 1 päiväntasaaja sekä leveyspiireihin liittyvät kulmat on väritetty punaisella. Vastaavasti nollapituuspiiri ja pituuspiirejä koskevat kulmat ovat sinisiä. Nollapituuspiiri kulkee Lontoossa sijaitsevan Greenwichin läpi.

GPS-paikannus haltuun

Leveys- ja pituuspiiritiedon lisäksi GPS-moduulin avulla saadaan selville maanopeus ja arvioitu korkeus mitattuna merenpinnasta. Rajapinta tarjoaa myös funktiot esimerkiksi koordinaattipisteiden etäisyyden laskemiseen ja pisteen etsimiseen tietyltä alueelta. Qt:n sijaintirajapinta otetaan käyttöön QML:ssä seuraavalla käskyllä:

```
import QtPositioning 5.2
```

Rajapinnan tärkeimmät oliot ovat PositionSource, Position ja Location. Näiden lisäksi rajapintaan kuuluu coordinate-tyyppi paikkakoordinaatteja varten. PositionSource-olion avulla pyydetään sijaintitietoa, Position-olio esittää muuttuvan sijainnin ja Location-olio on tarkoitettu pysyvämpien paikkojen kuvaamiseen.

Esimerkiksi Position-oliolla on speed-ominaisuus nopeuden lukemista varten, mutta Location-oliolta se puuttuu. Location-oliolla ja sen ominaisuuksien avulla puolestaan voi tallentaa esimerkiksi postiosoitteen mutta Position-oliolla ei.

Käytettävissä olevat paikannusmenetelmät katsotaan *PositionSource*-olion supportedPositioningMethods-ominaisuuden avulla ja haluttu menetelmä valitaan asettamalla preferredPositioningMethods. Oletusarvoisesti käytetään kaikkia tarjolla olevia menetelmiä.

Sijaintipäivityksiä pyydetään tasaisin välein asettamalla päivitysväli (ominaisuus updateInterval) millisekunteina ja ominaisuus active arvoon true. Vaihtoehtoisesti voi pyytää vain yhtä sijaintipäivitystä kutsumalla update()-jäsenfunktiota eli metodia.

Kun sijaintitieto päivittyy, olio lähettää signaalin positionChanged(). Päivityshetken sijaintitieto on annettu *Position*-oliona, joka luetaan PositionSource-olion ominaisuudesta position. Position sisältää muun muassa ominaisuudet coordinate, speed ja horizontalAccuracy.

Ominaisuus speed kertoo lasketun maanopeuden, ja horizontalAccuracy antaa arvion sijaintitiedon tarkkuudesta metreinä. Ominaisuus coordinate puolestaan on tyypiltään *coordinate* ja sisältää pituuspiirin (longitude) ja leveyspiirin (latitude) arvot asteina sekä korkeuden (altitude) merenpinnasta metreinä.

Tyypin ”coordinate” olioita ei voi luoda itsenäisesti komennolla

```
coordinate { id: ... }
```

vaan se on aina tallennettava ominaisuuteen tai Javascript-muuttujaan. Rajapinnan mukaisia coordinate olioita voi luoda QtPositioning-olion funktiolla QtPositioning.coordinate(lat, lon).

Itse toteutetuissa olioissa käytetään var-tyypin ominaisuutta tallennukseen. Tyyppi *var* on QML:n yleistyyppi, johon voi tallentaa muita tyyppejä, esimerkiksi lukuja, merkkijonoja ja olioita. Se vastaa Javascriptin var-avainsanalla määritettäviä muuttujia.

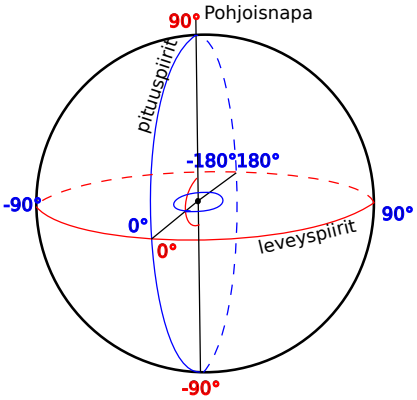
Tutkitaan GPS-koodia

Listauksessa 1 on paikannusesimerkin koodi. Koodilistauksissa käytetään kolmea pistettä (...) ilmaisemaan käsiteltävän asian kannalta epäoleellista koodia. Esimerkiksi käyttöliittymän painikkeet ja niiden käyttö on käsitelty Skrollin numerossa 2014.2, eikä niiden koodia tässä enää toisteta.

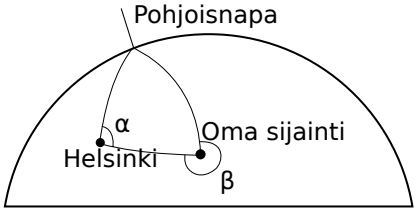
Paikannusesimerkissä käytetään käyttäjän määrittelemää päivitysväliä ja oletuspaikannusmenetelmiä. Nopeus ja korkeus näytetään puhelimen ruudulla, jos mittausarvot ovat saatavilla. Totuusarvoiset Valid-sanaan loppuvat ominaisuudet kertovat, sisälsikö mittauspäivitys kelvolliset arvot. Esimerkiksi jos speedValid-ominaisuus on arvoltaan tosi, päivitys sisältää nopeustiedon.

Lisäksi *Location*-oliolla esitetään Helsingin sijainti. Sen coordinate-ominaisuuden avulla voidaan laskea nykyisen sijainnin etäisyys Helsinkiin ja suuntima Helsingistä katsoen. Suuntima tarkoittaa siis Helsingistä pohjoisnavalle ja puhelimen sijaintiin piirrettyjen kaarten välistä kulmaa (kuva 2).

Etäisyys ja suuntima lasketaan coordinate-tyypin jäsenfunktioilla distanceTo(coordinate other) ja azimuthTo(coordinate other). Kummallekin annetaan parametrina toinen koordinaattipiste. Ensin mainittu laskee etäisyyden maapallon kehää pitkin, ja toinen laskee kulman myötäpäivään suhteessa maantieteelliseen pohjoisnapaan. Kumpikin olettaa maapallon täydelliseksi palloksi, eli maaston korkeuseroja eikä elliptisyyttä ei laskuissa huomioida.



Kuva 1. Koordinaattijärjestelmän määritelmät.



Kuva 2. Kulma α on suuntima Helsingistä ja kulma β suuntima Helsinkiin.

Esimerkkikoodissa kutsulla

```
helsinki.coordinate
  .distanceTo(
    positionSrc.position.coordinate)
```

lasketaan kuvassa 2 piirretyn Helsingin ja oman sijainnin välisen kaaren pituus. Kutsulla

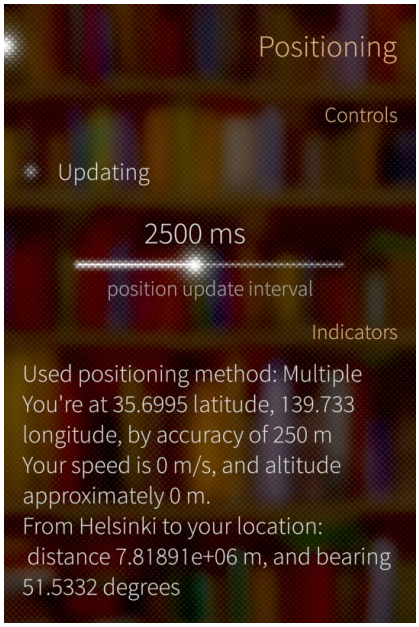
```
helsinki.coordinate
  .azimuthTo(
    positionSrc.position.coordinate)
```

puolestaan saadaan kuvassa näkyvä kulma α.

Kartan näyttäminen

Edellä opittiin selvittämään, missä Jolla seilaa. Katsotaan seuraavaksi, miten tämä tieto havainnollistetaan kartalla. Tätä varten Qt:ssa on *QtLocation*-moduuli. Se tarjoaa karttaolion Map sekä joukon olioita, joiden avulla kartalle voi piirtää. Rajapinnan olioiden avulla karttamerkinnät voi sitoa koordinaatteihin, jolloin merkinnät liikkuvat kartan mukana. Listauksessa 2 on kartan näyttämiseen ja käyttöön liittyvän sivun ohjelmakoodi.

Listauksen alussa on määritelty Drawer-olio, jonka avulla ruutu voidaan jakaa kartan ja käyttöliittymän kesken. Olion background-ominaisuuteen on laitettu SilicaFlickable, jonka lapsina ovat käyttöliittymän ohjaimet ja ohjetekstit. Jakautuminen näkyy kuvassa 4.



Kuva 3. Paikannussivu.

Listauksen keskivaiheilla on määritelty *Map*-olio, joka piirtää kartan ja tarjoaa valmiit eleet kartan selaamiseen, lähentämiseen ja loitontamiseen. Kartan koko asetetaan zoomLevel-ominaisuudella, jonka arvo on oletuksena 8. Suuremmalla arvolla piirretään pienempi alue tarkemmin näkyviin.

Kartta keskitetään asettamalla coordinate-tyypin ominaisuus center. Ensimmäisen kerran sen voi tehdä Component.onComplete-käsittelijässä, eli kun kartta on latautunut ja valmis.

Karttaoliolle on annettava lisäosaolio *Plugin*, joka kertoo, mitä karttapalvelua käytetään. Esimerkkiohjelmassa käytetään lisäosaa ”osm” eli Open Street Mapia. Lisäosille voi antaa parametreja lisäämällä niiden lapsioliooiksi PluginParameter-olioita.

Parametreilla voi hienosäätää lisäosan toimintaa, mutta ne jätetään käsittelemättä tässä artikkelissa. Ainoastaan rajapinta-avaimen antaminen Here-lisäosalle (`name: "nokia"`) näytetään. Nykyiset lisäosat toimivat vain online-tilassa, eli karttojen näyttäminen vaatii internet-yhteyden.

Karttamerkintöjen tekeminen

Edellä kerrotut ohjeet riittävät kartan näyttämiseen, mutta karttaan voi myös kiinnittää erilaisia QML-olioita rajapintaan kuuluvien Map-alkuisten olioiden avulla. Esimerkissä on käytetty MapQuickItem-, MapCircle- ja MapPolyline-olioita. MapMarker-olio ei kuitenkaan ole rajapinnan olio vaan oma komponenttinsa, jota käytetään MapQuickItemin kanssa.

Näiden lisäksi esimerkissä käytetään listauksessa 3 esitettyä MapModes-oliota. Tämäkin on sovelluksen oma olio, joka esittää käytettäviä karttatiloja. Listauksessa 2 on käytössä tilat drawRoute, placeMarkers ja markersNearby.

Kartalle lisätään olioita kutsumalla *Map*-olion addMapItem(item)-jäsenfunktiota. Vastaavasti poisto onnistuu kutsumalla removeMapItem(item)-funktiota. Lisättävän olion on oltava karttaolion lapsi. Tämä saavutetaan joko staattisesti QML:nä esitettynä (listaus 2, id:t ownPosition ja uncertaintyCircle) tai dynaamisesti antamalla karttaolio ensimmäisenä parametrina Component-olion createObject(parent, properties)-jäsenfunktiolle (listaus 2, addMarker()-funktio).

Lisäksi kartta tarjoaa muunnokset ruudun xy-pisteen muuttamiseksi koordinaattipisteeksi ja päinvastoin. Nämä funktiot ovat nimeltään toCoordinate(point) ja fromCoordinate(coordinate). Ensimmäisen parametrina käytetyn olion voi luoda funktiolla Qt.point(x, y) ja jälkimmäisen funktiolla QtPositioning.coordinate(lat, lon).

Listauksen 2 keskivaiheilla Map-olion lapsina ovat MapQuickItem, MapCircle ja MapPolyline. Näiden kolmen käyttö selitetään seuraavaksi.

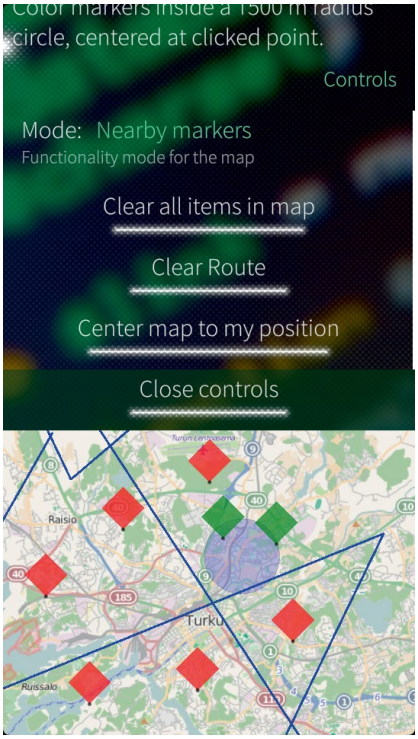
MapQuickItem on yleiskäyttöisin piirrettävä karttaolio. Sille annetaan sourceItem-ominaisuudella toinen olio, joka piirretään kartalle coordinate-koordinaattiin. Koordinaatti ja piirrettävän olion piste sidotaan antamalla anchorPoint.x ja anchorPoint.y. Esimerkissä on luotu MapQuickItem-olio, jonka ominaisuus id on kuvattavan asian mukaisesti ownPosition. Oliossa sidotaan pallo keskipisteestään puhelimen GPS-paikantimelta saatuun sijaintiin.

MapCircle-oliolla piirretään kartalle ympyröitä. Esimerkissä on piirretty GPS:n ilmoittama epävarmuusalue tämän olion avulla. Piirtämistä varten ympyrän keskipisteen sijainti kerrotaan asettamalla ominaisuuteen center coordinate-tyypin olio. Ympyrän sädettä metreinä säädetään ominaisuuden radius kautta. Lisäksi ympyrän väriä, reunusta ja muita yksityiskohtia voi säätää, kuten esimerkissä on tehty.

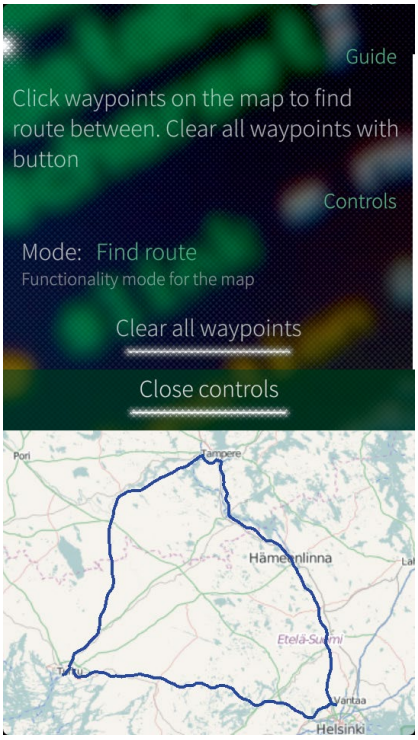
MapPolyline on tarkoitettu polkujen piirtämiseen. Kauttakulkupisteitä lisätään ja poistetaan jäsenfunktioilla addCoordinate(coord) ja removeCoordinate(coord). Kaikki pisteet voi poistaa asettamalla pisteitä sisältävä lista eli ominaisuus path arvoon []. Poistaessa yksittäisiä pisteitä on parametrina annettava jokin coordinate-olio, joka kuuluu listaan path. Uuden coordinate-olion luominen samaan pisteeseen ei riitä.

Listauksen 2 lopussa esiteltäviä apufunktioita kutsutaan, kun karttaa kosketetaan klikkauksenomaisesti. Kutsuttava funktio riippuu valitusta karttatilasta. Valinta tapahtuu switch–case-rakenteella MouseArea-olion onClicked-käsittelijässä, joka on muutamia rivejä funktioiden määrittelyjä ylempänä.

Funktio showMarkersNear(point) värittää 1500 metrin säteisen, koskettuun pisteeseen keskitetyn ympyrän sisään jäävät karttamerkinnät vihreiksi. Piirrettävään polkuun (MapPolyline-olio) lisätään koordinaattipiste addRoutePoint(point)-funktiolla. Viimeinen funktio addMarker(point) luo uuden MapMarker-olion dynaamisesti ja lisää sen kartalle.



Kuva 4. Kartalle piirtelyä.



Kuva 5. Reititys- ja paikkasivu.

Malli–näkymä-ajattelun hyödyntäminen

Rajapinta tarjoaa myös QML:lle tyypillistä malli–näkymä-ajattelua varten MapItemView-olion. Tälle oliolle voi antaa mallin, esimerkiksi ListModel-olion, ja edustajan (delegate), joka on jokin kartalle piirrettäväksi kelpaava olio. MapQuickItemin avulla voidaan kuitenkin käyttää mitä tahansa omaa piirrettävää oliota, vaikkapa kuvaa tai tekstiä.

Menetelmän hyödyt huomaa helposti rajapinnan tarjoamien mallien avulla. Niitä käytetään reittien, paikkojen ja geokoodauksen hakemiseen. Katsotaan esimerkkinä reittien ja paikkojen haku rajapinnan malleja käyttäen.

Listauksen 4 esimerkkisivu on melko samanlainen kuin ensimmäinen karttasivu. Karttaolion Map lapsina on nyt kuitenkin kolme MapItemView-oliota. Kahta ensimmäistä käytetään reittimallin eli RouteModel-olion datan visualisoimiseen. Kolmas MapItemView näyttää PlaceSearchModel-olion eli paikkamallin tiedot. Edustajana on itse toteutettu olio PlaceMarker, joka pohjautuu MapQuickItemiin.

Reittimallille tarvittiin poikkeuksellisesti kaksi näkymäoliota, koska OSM-reititys on nykyisessä lisäosaversiossa (5.2.1+git6-1.22.6) rikki. Tästä syystä se tarvitsi oman edustajaolion, OSM-Routen luomista, jossa virhe (koordinaatit dekadia liian suuret) korjataan. Ehjällä lisäosalla voi käyttää suoraan rajapinnan valmista MapRoute-oliota.

Reittien etsiminen

Reitit löytyvät listauksen 4 loppuosassa käytetyn *RouteModel*-olion avulla. Sille täytyy asettaa ominaisuudet plugin ja query. Ensimmäiselle riittää viittaus Map-oliolle luotuun Plugin-olioon, mutta myös eri lisäosaa voi käyttää. Tuen reititykselle voi tarkistaa Pluginin jäsenfunktiolla supportsRouting(). Ilman parametreja funktio palauttaa arvon tosi, jos jotakin reititystapaa tuetaan.

Toiseen ominaisuuksista asetetaan *RouteQuery*-olio. Tällä oliolla on lukuisia ominaisuuksia reittihaun hienosäätämiseksi, kieltoalueista kulkuneuvon valintaan. Tärkeintä on kuitenkin reittipisteiden asettaminen, ja esimerkissä kaikki muut arvot on jätetty oletuksiin.

Reittipisteitä lisätään kutsumalla RouteQueryn addWaypoint(coordinate)-jäsenfunktiota. Esimerkin lopussa, Map-olion jälkeen, on esitelty apufunktio pisteen lisäämiselle. Funktio addRoutePoint(point) ottaa näytöltä kosketetun pisteen ja muuttaa sen koordinaatiksi. Piste lisätään reittikyselyyn addWaypoint-funktiolla. Lopuksi malli päivitetään, jos pisteitä on enemmän kuin kaksi.

Malli päivitetään kutsumalla RouteModelin jäsenfunktiota update(). Mikäli reitti löytyi kyselyn perusteella, päivittynyt data esitetään kartalla automaattisesti malli–näkymä-periaatteen mukaisesti.

Mallin tietoihin pääsee edustajassa käsiksi viittaamalla mallin rooliin routeData, joka on Route-olio. Olion voi sijoittaa suoraan MapRoute-olion route-ominaisuuteen, kuten listauksen 4 keskivaiheilla on tehty (`id: normalRoute`). Itse Route-olion yksityiskohdat selviävät dokumentaatiosta helposti, mutta mainittakoon, että tiedot ovat luettavissa ainoastaan olion ominaisuuksien kautta.

Paikkojen hakeminen

Paikkoja – kuten vaikka pitserioita, autokatsastamoita tai elokuvateattereita – voi etsiä paikkarajapinnan avulla, mikäli lisäosa tarjoaa sellaisen palvelun. Valitettavasti OSM-lisäosaan tätä ei ole toteutettu, mutta onneksi Here-palvelu tukee myös paikkojen hakua. Paikkoja haetaan PlaceSearchModel-olion avulla. Minimaaliset asetukset paikkojen hakemiseksi ilman rajoituksia ovat lisäosa, alue ja hakutermi.

Esimerkin *PlaceSearchModel* on listauksen 4 lopussa, RouteQueryn yläpuolella. Siinä on käytetty lisäosana Here-karttoja, jolloin Pluginin name-ominaisuudeksi tulee ”nokia”. On myös lisättävä listauksessa 4 kommentoidut kaksi PluginParameter-riviä, joissa asetetaan sovellusavain ja tunnus. Nämä ovat saatavissa ilmaisillakin käyttöehdoilla osoitteesta developer.here.com.

Hakualue sijoitetaan ominaisuuteen searchArea, ja sen tulee olla tyypiltään geoshape. Esimerkissä on luotu geocircle kutsumalla funktiota QtPositioning.circle(center, radius). Hakutermi searchTerm on pelkkä merkkijono. Muutosten jälkeen hakutulokset päivitetään kutsumalla jäsenfunktiota update().

Malli tarjoaa edustajalle useita eri rooleja, mutta esimerkissä on käytetty ainoastaan place-roolia. Tämä viittaa Place-olioon, joka sisältää muun muassa paikan nimen, tietoja ikonista ja Location-olion, jossa on paikan osoite ja koordinaatit.

Kohti tuntemattomia vesiä

Artikkelissa käytiin läpi runsas tietopaketti Qt:n paikannus- ja sijaintirajapinnoista. Ensin katsottiin, miten GPS-koordinaatit saa Jollasta ulos ja omaan ohjelmaan, sitten piirrettiin kartalle ja lopuksi tutkittiin valmiiksi toteutettuja, järjestelmällisiä tapoja yleisten navigointitehtävien tekemiseen eli reittien ja paikkojen hakemiseen. Näillä eväillä lukija voinee käyttää online-karttoja omissa sovelluksissaan, kuten jo jotkin sovellukset tekevät, esimerkiksi reitin nauhoitin ja paikallisbussien seuraamissovellus.

Esitellyille ominaisuuksille riittää myös sovelluskohteita, joita Sailfishille ei vielä ole tehty: geokätköilysovellus, salamatutka tai vaikka uutistenlukija, joka näyttää tapahtumapaikat kartalla. Mikäli innostus ei vielä riitä kokonaisen oman sovelluksen tekemiseen, esimerkkiohjelmaa kokeilemalla ja muokkaamalla pääsee kuitenkin tutustumaan perustoimintoihin. Koodausintoa kaikille!

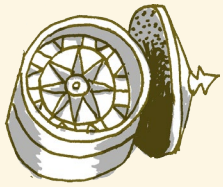
Navigoi lisätiedon lähteille

- developer.ubuntu.com/api/apps/qml/sdk-15.04/QtLocation/ – QtLocation-dokumentaatio. Sisältää Map-, Map-alkuisten, Place- ja malliolioiden tarkat ominaisuuslistat. Ubuntu-developerin dokumentaatio on jäädytetty versioon 5.0/5.2. Uudempi dokumentaatio löytyy doc.qt.io-osoitteesta.
- doc.qt.io/qt-5/qtpositioning-index.html – QtPositioning dokumentaatio. Sisältää tarkemmat tiedot Position-, PositionSource- ja Location-olioista.
- developer.ubuntu.com/api/apps/qml/sdk-15.04/QtLocation.location-plugin-nokia/ – Here-karttalisäosan käyttöohjeet. Osa esitellyistä parametreista toimii myös OSM-lisäosan kanssa.
- developer.ubuntu.com/api/apps/qml/sdk-15.04/QtLocation.location-plugin-osm/ – OSM-lisäosan parametrit ja käyttöohjeet.
- developer.ubuntu.com/api/apps/qml/sdk-15.04/QtLocation.Plugin/ – Plugin-olion dokumentaatio.
- doc.qt.io/qt-5/location-places-qml.html – Paikkarajapinta, jonka avulla voi etsiä karttapalveluun kirjattuja paikkoja.
- doc.qt.io/qt-5/qml-qtlocation-place.html – Yksi PlaceSearchModelin tarjoama olio paikkatiedoille.
- www.navis.fi/sanasto.html – Helsingin Navigaattorit ry:n navigointisanasto.

```
import QtQuick 2.0

QtObject {
  id: root

  property int ownPosition: 1
  property int drawRoute: 2
  property int placeMarkers: 3
  property int markersNearby: 4

  property int routing: 5
  property int placesNearby: 6

  function getDescription(mode)
  {
    switch ( mode )
    {
    case root.ownPosition:
      return qsTr("Press the map to show/hide full map,"
                  + " own position is shown with"
                  + " uncertainty circle.");
      ...
    }
  }

  function getMenuString(mode)
  {
    switch (mode)
    {
    case root.ownPosition:
      return  qsTr("Own position");
      ...
    }
  }

  property ListModel availableModes: ListModel {  }

  function addAvailableMode(mode)
  {
    availableModes.append({"modelMode": mode});
  }
}
```

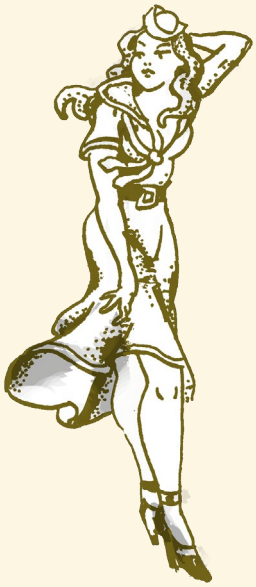
Listaus 3. MapModes-komponentti.

```
import QtQuick 2.0
import Sailfish.Silica 1.0

import QtPositioning 5.2

Page {
  id: root
  PositionSource {
    id: positionSrc
    updateInterval: intervalSlider.value
    active: activationSwitch.checked
  }

  Column {
    ...

    Slider {
      x: Theme.paddingLarge
      width: parent.width-2*Theme.paddingLarge
      id: intervalSlider
      label: qsTr("position update interval")
      minimumValue: 500
      maximumValue: 5000
      stepSize: 1
      value: 2500
      valueText: qsTr("%1 ms").arg(value)
    }

    SectionHeader {
      text: qsTr("Indicators")
    }

    Label {
      x: Theme.paddingLarge
      width: parent.width-2*Theme.paddingLarge
      wrapMode: Text.Wrap
      text: qsTr("Used positioning method: %1").arg(methodName)
      property string methodName: {
        switch(positionSrc.supportedPositioningMethods)
        {
        case PositionSource.SatellitePositioningMethods:
          return qsTr("Satellite");
        case PositionSource.NoPositioningMethods:
          return qsTr("Not available");
        case PositionSource.NonSatellitePositioningMethods:
          return qsTr("Non-satellite");
        case PositionSource.AllPositioningMethods:
          return qsTr("Multiple");
        default:
          return qsTr("source error");
        }
      }
    }

    Label {
      id: positionLabel
      x: Theme.paddingLarge
      width: parent.width-2*Theme.paddingLarge
      wrapMode: Text.WordWrap
      text: qsTr("You're at %1 latitude, %2 longitude,"
                 + " by accuracy of %3 m")
      .arg(positionSrc.position.coordinate.latitude)
      .arg(positionSrc.position.coordinate.longitude)
      .arg(positionSrc.position.horizontalAccuracy);
    }

    Label {
      id: speedLabel
      x: Theme.paddingLarge
      width: parent.width-2*Theme.paddingLarge
      wrapMode: Text.WordWrap
      text: qsTr("Your speed is %1 m/s,"
                 + " and altitude approximately %2 m.")
        .arg(positionSrc.position.speedValid
             ? positionSrc.position.speed : 0 )
        .arg(positionSrc.position.altitudeValid
             ? positionSrc.position.coordinate.altitude : 0);
    }

    Location {
      id: helsinki
      coordinate {
        latitude: 60.1708
        longitude: 24.9375
      }
    }
    Label {
      id: distanceLabel
      x: Theme.paddingLarge
      width: parent.width-2*Theme.paddingLarge
      wrapMode: Text.WordWrap
      text: qsTr("From Helsinki to your location:\n"
                 + " distance %1 m, and bearing %2 degrees")
        .arg(helsinki
          .coordinate.distanceTo(
            positionSrc.position.coordinate))
        .arg(helsinki
          .coordinate.azimuthTo(
            positionSrc.position.coordinate));
    }
  }
}
```



Listaus 1. Paikannussivun koodi.

```
import QtQuick 2.0
import Sailfish.Silica 1.0

import QtPositioning 5.2
import QtLocation 5.0

import "../components"

Page {
  id: root

  property var places: [];

  backNavigation: drawer.open

  MapModes {
    id: modes
  }

  PositionSource {
    id: positionSrc
    updateInterval: 1000
    active: true
  }

  Drawer {
    id: drawer
    ...

    background: SilicaFlickable {
      ...
      Column {
        ...


        ComboBox {
          id: modeSelector

          anchors.horizontalCenter: parent.horizontalCenter

          label: qsTr("Mode: ")
          description: qsTr("Functionality mode for the map")

          menu: ContextMenu {
            Repeater {
              model: modes.availableModes

              MenuItem {
                text: modes.getMenuString(modelMode)
                property int mode: modelMode
              }
            }
          }
        }

        Button {
          id: clearButton
          ...
        }
        Button {
          id: clearRouteButton
          ...
        }
        Button {
          id: centerButton
          ...
        }
      }
    }

    Column {
      id: mapContent
      ...
      Button {
        id: controlsButton
        anchors.horizontalCenter: parent.horizontalCenter
        text: drawer.open
            ? qsTr("Close controls") : qsTr("Open controls")
        onClicked: {
          drawer.open = !drawer.open;
        }
      }

      Map {
        id: map
        ...
        zoomLevel: 10

        plugin: Plugin {
          name: "osm"
        }

        MapQuickItem {
          id: ownPosition

          anchorPoint.x: positionCircle.width / 2
          anchorPoint.y: positionCircle.width / 2

          coordinate: positionSrc.position.coordinate

          sourceItem: Rectangle {
            id: positionCircle
            width: 6
            height: width
            radius: width
            color: "blue"
          }
        }

        MapCircle {
          id: uncertaintyCircle

          center: positionSrc.position.coordinate
          radius: positionSrc.position.horizontalAccuracy

          color: "blue"
          opacity: 0.3

          border.color: "blue"
          border.width: 2
        }

        MapCircle {
          id: containingCircle
          ...
        }

        MapPolyline {
          id: route

          line.color: "blue"
          line.width: 3
        }

        MouseArea {
          anchors.fill: parent
          onClicked: {
            switch ( modeSelector.currentItem.mode )
            {
              case modes.drawRoute:
                addRoutePoint(Qt.point(mouse.x, mouse.y));
                break;
              case modes.placeMarkers:
                addMarker(Qt.point(mouse.x, mouse.y));
                break;
              case modes.markersNearby:
                showMarkersNear(Qt.point(mouse.x, mouse.y))
                break;
              default:
                break;
            }
          }
        }
        Component.onCompleted: {
          center = QtPositioning.coordinate(60.45, 22.25);

          modes.addAvailableMode(modes.drawRoute);
          modes.addAvailableMode(modes.placeMarkers);
          modes.addAvailableMode(modes.markersNearby);

          map.addMapItem(ownPosition);
          map.addMapItem(uncertaintyCircle);
          map.addMapItem(route);
        }
      }
    }
  }

  function showMarkersNear(point)
  {
    containingCircle.center = map.toCoordinate(point);
    var circle = QtPositioning.circle(containingCircle.center,
                                      containingCircle.radius);
    places.forEach(function (place,index,originalList)
    {
      if(circle.contains(place.coordinate))
      {
        place.sourceItem.color = "green";
      }
      else
      {
        place.sourceItem.color = "red";
      }
    });
  }

  function addRoutePoint(point)
  {
    route.addCoordinate(map.toCoordinate(point));
  }

  function addMarker(point)
  {
    var marker = locationMarkerComponent
        .createObject(
          map,
          {coordinate: map.toCoordinate(point)}
        );
    map.addMapItem(marker);
    places.push(marker);
  }

  Component {
    id: locationMarkerComponent
    MapMarker {
    }
  }
}
```



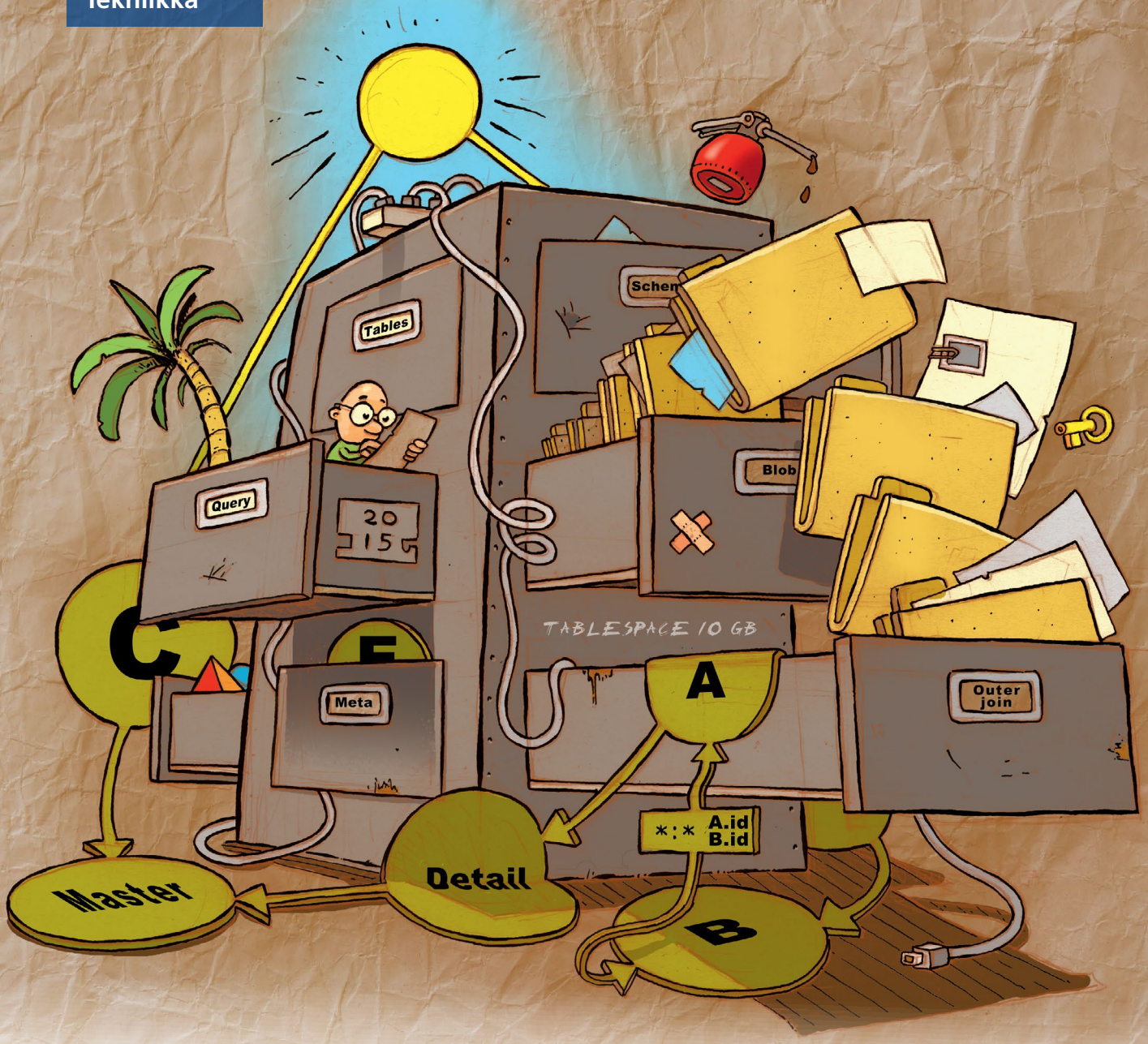
Listaus 2. Karttasivu.

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import QtPositioning 5.2
import QtLocation 5.0

import "../components"

Page {
 id: root

 backNavigation: drawer.open

 MapModes {
   id: modes
 }

 Drawer {
   id: drawer
   ...
   background: SilicaFlickable {
     ...
     Column {
       id: controls
       ...

       ComboBox {
         id: modeSelector
         ...
       }

       TextField {
         id: keywordField
         ...
       }
       Button {
         id: clearWaypointsButton
         ...
       }
     }
   }

   Column {
     id: mapContent
     ...

     Map {
       id: map
       ...
       plugin: Plugin {
         id: mapPlugin
         name: "osm"
         Component.onCompleted: {
           if(mapPlugin.supportsRouting())
           {
             modes.addAvailableMode(modes.routing);
           }
           if(mapPlugin.supportsPlaces())
           {
             modes.addAvailableMode(modes.placesNearby);
           }
         }
       }
       MapItemView {
         model: routeModel
         delegate: Component {
           id: normalRoute
           MapRoute {
             visible: mapPlugin.name !== "osm"

             route: routeData

             line.color: "blue"
             line.width: 4
             smooth: true
             opacity: 0.9
           }
         }
       }
       MapItemView {
         model: routeModel
         delegate: Component {
           // Due to a bug, OSM routing gives
           // coordinates too large by a decade.
           // OSMRoute fixes the coordinates
           // and draws the route with MapPolyline
           OSMRoute {
             visible: mapPlugin.name === "osm"
             route: routeData
           }
         }
       }
       MapItemView {
         model: searchModel
         delegate:  Component {
           PlaceMarker {
             name: place.name
             icon: place.icon
             coordinate: place.location.coordinate
           }
         }
       }
       MouseArea {
         ...
         onClicked: {
           switch ( modeSelector.currentItem.mode )
           {
           case modes.routing:
             addRoutePoint(Qt.point(mouse.x, mouse.y));
             break;
           case modes.placesNearby:
             showPlacesNear(Qt.point(mouse.x, mouse.y),
                                     keywordField.text)
             break;
           default:
             break;
           }
         }
       }
       Component.onCompleted: center = QtPositioning.
         coordinate(60.45, 22.25);
     }
   }
 }

 function showPlacesNear(point, keyword)
 {
   searchModel.searchArea
     = QtPositioning.circle(map.toCoordinate(point),
                                             1500);
   searchModel.searchTerm = keyword;
   searchModel.update();
 }

 function addRoutePoint(point)
 {
   routeQuery.addWaypoint(map.toCoordinate(point));
   if(routeQuery.waypoints.length >= 2)
   {
     routeModel.update();
   }
 }

 PlaceSearchModel {
   id: searchModel
   plugin: Plugin {
     name: "nokia"
     // set the correct values and uncomment the following to use
     // places:
     // PluginParameter{ name: "app_id" ; value: "your_appid" }
     // PluginParameter{ name: "token" ; value: "your_token" }
     Component.onCompleted: {
       if(supportsPlaces() && appIdAndTokenGiven())
       {
         modes.addAvailableMode(modes.placesNearby);
       }
     }

     function appIdAndTokenGiven()
     {
       ...
       return appIdGiven && tokenGiven;
     }
   }
   property bool isLoading: status === PlaceSearchModel.Loading
 }

 RouteQuery {
   id: routeQuery
 }

 RouteModel {
   id: routeModel
   plugin: mapPlugin
   query: routeQuery
   property bool isLoading: status === RouteModel.Loading
 }
 ...
}
```

Listaus 4. Reittien ja paikkojen hakusivu.

Näyttääkö tutulta?

doop.fi/opensource-teamleader/
doop.fi/drupal-developer/



Tietoa kantaan ja takaisin

Tietokannat ovat tietokonealan perustekniikkaa, jota ilman pyörät eivät pyöri eikä bitti etene. Mutta mitä tietokannat oikeastaan ovat, ja miten ne toimivat? Nyt otetaan selvää.

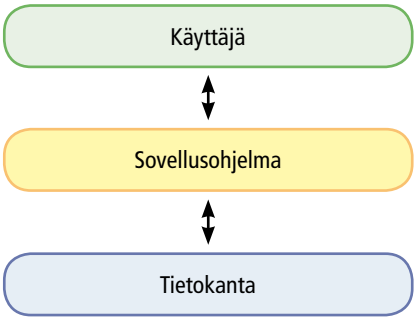
Teksti: Teemu Likonen
Kuvat: Tapio Lehtimäki, Teemu Likonen, Mitol Meerna, Nasu Mäki-Petäys

Tietokanta on mikä tahansa järjestelmällinen tietovarasto, eikä se välttämättä edes liity tietokoneisiin mitenkään. Esimerkiksi puhelinluettelo, kalenteri ja www-palvelimen lokitiedosto ovat tietokantoja, koska niihin tallennetaan tietoa säännönmukaisessa muodossa.

Nykyaikana tietokantoja on joka paikassa ja ne palvelevat hyvin erilaisia tarpeita. Ilmeisimpiä käyttökohteita ovat kirjaston tiedonhakupalvelu sekä väestö- tai asiakasrekisterin kaltaiset henkilötietokannat. Myös kaupalliset yhteisöpalvelut kuten Facebook ja Twitter voidaan ajatella käyttöliittymänä valtavan suureen tietokantaan, johon on tallennettu käyttäjätiedot, viestit ja kaikki muukin sisältö.

Monet tietokannat ovat piilossa tai niin pieniä, ettei niitä edes miellä tietokannoiksi. Esimerkiksi nettiselaimen suosikkilinkit ja selaushistoria ovat ohjelman omassa tietokannassa. Myös tietokonepelin hahmot ja pelimaailman tilanne tallennetaan tietokantaan, joka on piilossa peliohjelman sisällä.

Tietokannat ovat eräänlaista pinnan alla vaikuttavaa tekniikkaa, jonka varaan monet muut ohjelmat ja tekniikka rakentuvat. Siksi niiden ymmärtäminen kuuluu tietotekniikan yleissivistykseen. Tässä yhteydessä emme sukella ohjelmakoodin sekaan vaan otamme haltuun tietokantojen



Tietotekniikan ravintoketju.

peruskäsitteet sekä tärkeimpiä toimintaperiaatteita.

Perusarvona järjestelmällisyys

Ei ole aivan yhdentekevää, kuinka varastoi arvokkaat tietonsa tietokoneelle. Hyvin järjestetystä tietokannasta asiat löytyvät nopeasti ja asioiden välille on helppo muodostaa yhteyksiä. Säännönmukaista tietokantaa on myös paljon mukavampi ylläpitää kuin sekavaa bittitunkiota.

Järjestelmällisyys tarkoittaa käytännössä esimerkiksi sitä, että kaikki kantaan lisättävä tieto tarkistetaan huolellisesti ja muutetaan tiettyyn yhtenäiseen muotoon. Näin tietokantaan ei pääse lipsahtamaan vääränlaista tietoa, joka voisi sekoittaa ohjelman toiminnan. Rikkoutuneen tietokannan korjaaminen saattaa olla vaikeaa, joten on parempi ehkäistä ongelmat jo ennalta.

Onneksi tietokantojen rakentelu ei ole kovin vaikeaa rakettitiedettä. Kyse on vain johdonmukaisesta, yhtenäisestä järjestelmästä. Ihan pelkällä arkiajattelulla ei kuitenkaan pärjää, sillä on pystyttävä ajattelemaan tietovarastoa tietokoneen kannalta. Mikä on tietokoneelle luonteva tapa käsitellä tietoa? Yksi vastaus ovat taulukot ja niiden väliset suhteet; toinen vastaus ovat dokumentit ja niihin kytketyt metatiedot. Perehdytään ensin taulukoihin.

Tiedot taulukkoon

Suurten tietomäärien varastointi perustuu yleensä taulukoihin. Taulukon sarakkeilla on otsikko, esimerkiksi ”Sukunimi”, ja samaan sarakkeeseen tallennetaan ainoastaan samantyyppistä tietoa. Sarakkeita ja niihin kuuluvia soluja kutsutaan myös *kentiksi* (field). Taulukon rivit puolestaan ovat *tietueita* (record), jotka kuvaavat jotakin yhtä tietokokonaisuutta, esimerkiksi yhden henkilön tietoja. (Esimerkki 1.)

Taulukossa on yleensä ensimmäisenä tunnistesarake, joka on tässä esimerkissä kasvava kokonaisluku: 1, 2, 3 jne. Tunnisteen tehtävänä on yksilöidä jokainen rivi eli tietue. Tietokannassa voi olla esimerkiksi samannimisiä henkilöitä, mutta tunnisteen avulla tietokone osaa silti erottaa ne toisistaan.

Taulukkorakenne määrittelee tarkasti, millaista tietoa tietokantaan voidaan tallentaa. On tavallista, että tietokantaan määritellään sarakkeille tietotyyppi, niin että esimerkiksi Sukunimi-kenttään hyväksytään vain merkkijonotyyppinen sisältö ja Syntymäaika-kenttään vain päivämäärä.

Taulukon rakennetta ei saa sotkea esimerkiksi siten, että tallentaa yhteen Puhelin-kenttään saman henkilön kaksi eri puhelinnumeroa pilkulla erotettuna. Ihminen toki ymmärtää, mistä on kyse, mutta tietokoneen kannalta sisällön käsittely vaikeutuu. Mistä kone tietää, että yksi kenttä sisältääkin kaksi eri puhelinnumeroa? Tietokoneohjelmalle voisi kyllä opettaa, että pilkku erottaa numerot toisistaan, mutta osataanko käytäntöä noudattaa järjestelmällisesti? Joudutaan ehkä ohjelmoimaan poikkeuksellisia tarkistus- ja tiedonhakutoimintoja, jotta tietokanta toimii halutulla tavalla. Yleensä tällaiseen leikkiin ei kannata ryhtyä vaan on hyvä sijoittaa yhteen kenttään vain yhdenlainen tieto.

Taulukot suhteessa toisiinsa

Edellä esitelty henkilötietokanta sallii henkilöä kohti vain kaksi puhelinnumeroa. Yhteystietojen kannalta se on kovin kankea, koska todellisuudessa ihmisillä on usein monenlaisia sähköisen viestinnän yhteystietoja kuten sähköpostiosoitteita, Twitter-tunnus ja blogi. Joustavampi tietokanta saadaan luomalla yhteystietoja varten kokonaan oma taulukkonsa ja kytkemällä se toiseen taulukkoon, joka sisältää henkilöiden nimet.

Esimerkistä 2 selviää, kuinka kaksi taulukkoa kytketään toisiinsa. Henkilöiden nimet ovat yhdessä taulukossa ja yhteystiedot toisessa. Yhteystiedot-taulukko sisältää kentän ”Henkilön tunniste”, jossa oleva luku viittaa Henkilöt-taulukon Tunniste-kenttään. Näin tietokanta osaa kytkeä henkilöille erilaisia yhteystietoja ja niitä voi olla miten paljon tahansa. Tietokannan termeillä ilmaistuna taulukoiden välillä on *yhdestä moneen -suhde*, koska yhden taulukon yhtä tietuetta kohti voi olla useita tietueita toisessa taulukossa.

Taulukoiden väliset suhteet eli *relaatiot* ovat monien tietokantojen keskeinen piirre. Puhutaankin relaatiotietokannoista, kun halutaan viitata nimenomaan tällaisiin tietokantoihin. Ennen kuin jatkamme eteenpäin, on syytä esitellä pari muutakin relaatioihin liittyvää käsitettä ja termiä.

Taulukossa olevaa yksilöivää tunnistesaraketta kutsutaan taulukon *perusavaimeksi* (primary key). Se on yksilöllinen tunnus, johon tyypillisesti viitataan muista taulukoista. Perusavain ei välttämättä muodostu yhdestä tunnistesarakkeesta, vaan joissakin tilanteissa esimerkiksi kaksi saraketta yhdessä muodostavat perusavaimen, joka on yksilöllinen jokaiselle tietueelle.

Yhden taulukon perusavaimeen viittaava toisen taulukon sarake on nimeltään *viiteavain* (foreign key). Edellä käsitellyssä esimerkissä sarake ”Henkilön tunniste” toimii viiteavaimena, koska se viittaa toisen taulukon perusavaimeen.

Tietokantajärjestelmiin voi määrittää viiteavaimen ja perusavaimen välille riippuvuuksia: esimerkiksi jos tietokannasta poistetaan henkilötietue, poistuvat samalla automaattisesti kaik-

Tunniste	Sukunimi	Etunimi	Puhelin	Puhelin (työ)
1	Ruuskanen	Seija	040 123 4567	
2	Alaluusua	Markus	050 111 2222	016 999 8888
3	Leppänen	Timo	044 222 3333	

Esimerkki 1. Yksinkertainen henkilötietokanta, joka koostuu yhdestä taulukosta.

Taulukko: Henkilöt

Tunniste	Sukunimi	Etunimi
1	Ruuskanen	Seija
2	Alaluusua	Markus
3	Leppänen	Timo

Taulukko: Yhteystiedot

Tunniste	Henkilön tunniste	Yhteystapa	Yhteystieto
1	1	Puhelin	040 123 4567
2	2	Puhelin	040 111 2222
3	2	Puhelin (työ)	016 999 8888
4	3	Puhelin	044 222 3333
5	1	Twitter	@ruusseija
6	2	Sähköposti	markus@igi.fi
7	3	Sähköposti	timo@suomi99.fi

Esimerkki 2. Taulukot kytketään toisiinsa tunnistekenttien avulla.

ki kyseisen henkilön yhteystiedot toisesta taulukosta. Tämä on hyödyllinen ominaisuus, jonka avulla tietokanta siivoaa automaattisesti turhat tietueet pois eli huolehtii omasta eheydestään.

Monesta moneen

Esimerkkinä ollut kahden taulukon henkilötietokanta on ihan toimiva ja käyttökelpoinen, ainakin henkilökohtaiseen käyttöön, mutta vaativampiin tarpeisiin se on vähän kömpelö. Se nimittäin rikkoo yhtä tärkeää tietokantasuunnittelun periaatetta: samaa tietoa ei pitäisi toistaa monessa paikassa.

Yhteystietoja sisältävässä taulukossa (esimerkki 2) on sarake ”Yhteystapa”, jossa toistuu useita kertoja sana ”Puhelin”. Laajassa henkilötietokannassa voi helposti olla vaikkapa miljoona puhelinnumeroa, ja olisi tuhlausta kirjoittaa Puhelin-sana niin monta kertaa.

Saman asian toistosta voi olla muutakin haittaa. Kun ihminen syöttää henkilön puhelinnumeroita tietokantaan, hän saattaa eri kerroilla kirjoittaa yhteystavan nimen eri tavoin. Lopulta yhden henkilön kohdalla voi lukea ”Puhelin (työ)”, toisen kohdalla ”Puhelin, työ” ja kolmannella ”Työpuhelin”. Ihminen ymmärtää, että kaikki kolme ilmausta tarkoittavat samaa, mutta on hankalampaa opettaa tietokone löytämään kannasta kaikki työpuhelinnumerot.

Ongelmat ratkaistaan määrittelemällä yhteystavat omassa taulukossaan ja jokainen vain yhden kerran. Yhteystietoja sisältävään taulukkoon tehdään viiteavain ”Yhteystavan tunniste”, joka viittaa Yhteystavat-taulukossa olevaan tunnisteeseen. Esimerkki 3 havainnollistaa uutta rakennetta.

Uudistettu henkilötietokanta esittelee uudenlaisen suhteen taulukoiden välillä. Henkilöt-taulukon ja Yhteystavat-taulukon välillä on *monesta moneen -suhde*, koska jokaisella henkilöllä voi olla useita yhteystapoja ja samanniminen yhteystapa voi olla usealla eri henkilöllä. Tällaista suhdetta ei voi ilmaista kahden taulukon välillä suoraan, vaan välittäjäksi tarvitaan aina kolmas taulukko, joka sisältää viiteavaimet kahteen muuhun taulukkoon. Tässä tapauksessa Yhteystiedot-taulukko toimii välittäjänä eli kytkee henkilöt ja erilaiset yhteystavat toisiinsa.

Monesta moneen -suhdetta tarvitaan myös esimerkiksi kirjaston tietokannassa, jossa muun muassa kytketään teoksia ja tekijöitä toisiinsa. Yhdellä teoksella voi olla useita tekijöitä, ja yksi tekijä voi olla julkaissut useita teoksia. Esimerkissä 4 on kirjaston tietokanta pelkistettynä. Se sisältää perustaulukot ja -relaatiot teosten, asiakkaiden ja lainausten käsittelyyn. Harjoitustehtävänä voisi miettiä, miten siihen lisätään teosten varausmahdollisuus tai tekijöiden erilaiset vastuualueet teoksessa: kirjailija, toimittaja, säveltäjä, sanoittaja jne.

Taulukko: Henkilöt

Tunniste	Sukunimi	Etunimi
1	Ruuskanen	Seija
2	Alaluusua	Markus
3	Leppänen	Timo

Taulukko: Yhteystavat

Tunniste	Yhteystapa
1	Puhelin (oma)
2	Puhelin (työ)
3	Twitter
4	Sähköposti

Taulukko: Yhteystiedot

Tunniste	Henkilön tunniste	Yhteystavan tunniste	Yhteystieto
1	1	1	040 123 4567
2	2	1	040 111 2222
3	2	2	016 999 8888
4	3	1	044 222 3333
5	1	3	@ruusseija
6	2	4	markus@igi.fi
7	3	4	timo@suomi99.fi

Esimerkki 3. Toiston välttämiseksi yhteystapojen nimet on siirretty omaan taulukkoonsa. Henkilöiden ja yhteystapojen välillä on monesta moneen -suhde.

Tietokanta vastaa, kun siltä kysyy

Useista toisiinsa liittyvistä taulukoista koostuva tietokanta näyttää ihmisen kannalta sekavalta, koska tiedot ovat hajallaan ja pelkkiä lukuja sisältävät tunniste- ja viitekentät eivät ole kovin havainnollisia. Sen sijaan tietokoneelle ne ovat helppoa peruskauraa. Kone yhdistää eri taulukoiden tietueet nopeasti toisiinsa ja muodostaa mielekkäitä hakutuloksia. Tietokantajärjestelmiin sisältyy runsaasti valmiita tiedonkeruu- ja tilastotoimintoja.

Relaatiotietokannat tottelevat Structured Query Language (SQL) -kieltä, joka on juuri niitä varten suunniteltu kyselykieli. Kielen avulla paitsi haetaan tietoa mutta myös luodaan taulukoita sekä lisätään ja muokataan tietoa.

Tässä yhteydessä emme perehdy SQL-kieleen, mutta pari esimerkkiä lienee paikallaan. Esimerkissä 5 on kysely, joka hakee aiemmin esitellystä tietokannasta yhden henkilön kaikki yhteystiedot, ja esimerkissä 6 haetaan kannasta kaikki sähköpostiosoitteet. Kyselyn tulokset ovat kyselylauseen alla olevassa taulukossa.

SQL-kyselyjen kirjoittaminen ei tietenkään ole tavallisen käyttäjän hommaa, vaan ihmisen ja tietokannan välissä on sovellusohjelma, joka esimerkiksi tulkitsee käyttäjän tarpeet, muuttaa ne sopiviksi kyselyiksi ja näyttää tulokset ohjelman käyttöliittymässä.

Dokumentteja, ei taulukoita

Käytin aiemmin ilmausta ”bittitunkio” vertauskuvana huonosti järjestetylle ja vaikeasti ylläpidettävälle tietokannalle. Tiedon varastoinnissa kannattaakin pyrkiä järjestelmällisyyteen, mutta aina todellisuus ei jäsenny luontevasti taulukoihin ja niiden välisiin suhteisiin. Joskus täytyy varastoida sekalaista tavaraa, joka on epäyhtenäistä ja jota ei edes tunneta ennalta. Selvää ja loogista tietokantarakennetta ei välttämättä pystytä suunnittelemaan etukäteen.

Tämänkaltaiset tarpeet ovat synnyt-

Taulukko: Asiakkaat

Tunniste	Sukunimi	Etunimi
1	Lukija	Liisa
...	...	...

Taulukko: Luokat

Tunniste	Luokan nimi
84.2	suomenkielinen kertomakirjallisuus
...	...

Taulukko: Tekijät

Tunniste	Sukunimi	Etunimi
1	Sinisalo	Johanna
...	...	...

Taulukko: Teokset

Tunniste	Luokan tunniste	Teoksen nimi
1	84.2	Ennen päivänlaskua ei voi
...	...	...

Taulukko: Asiasanat

Tunniste	Asiasana
1	fantasiakirjallisuus
...	...

Taulukko: Lainat

Tunniste	Asiakkaan tunniste	Teoksen tunniste	Eräpäivä
...	...	...	...

Teoksen tunniste	Tekijän tunniste
...	...

Taulukko: Teokset + tekijät

Teoksen tunniste	Asiasanan tunniste
...	...

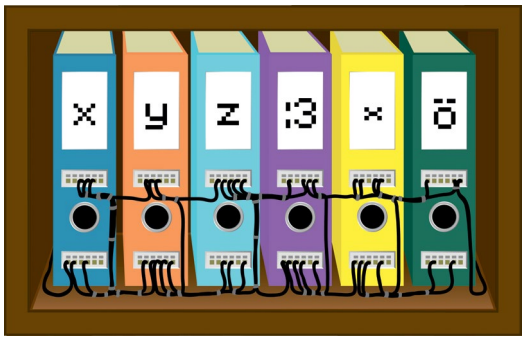
Taulukko: Teokset + asiasanat

Esimerkki 4. Kirjaston tietokanta pelkistettynä.

täneet uuden joukon tietokantajärjestelmiä, joissa varastoinnin perus-elementtinä eivät ole taulukot vaan dokumentit. Ajatuksena on, että tietokantaan voi tallentaa itsenäisiä, toisistaan poikkeavia dokumentteja eikä tietokannalle välttämättä määritellä mitään kokonaisrakennetta etukäteen. Rakenne voi muuttua tarpeiden mukaan joustavasti.

Dokumenttipohjainen tietokanta ei välttämättä tarkoita täydellistä epäjärjestystä. Dokumentit sinänsä ovat usein rakenteellisia kokonaisuuksia, jotka sisältävät myös metatietoa eli tietoa tiedosta. Metatieto kertoo esimerkiksi, että ”tässä on sähköpostiosoite” tai ”tässä on henkilön sukunimi”. Tietokannan hakutoiminnot hyödyntävät useimmiten juuri metatietoa. Voidaan esimerkiksi etsiä kaikista dokumenteista tiedot, jotka on varustettu metatiedolla ”Sähköposti”.

Suosituksi dokumenttien käsittelymuodoksi on noussut Javascript Object Notation (JSON), jolla voi kuvata metatietoa ja hierarkkisia tietorakenteita havainnollisesti – niin ihmisen kuin tietokoneenkin näkökulmasta. Esimerkiksi henkilön yhteystiedot voisi ilmaista JSON-muotoisena dokumenttina oheisen esimerkin 7 tavoin. Tässä tapauksessa dokumentti vastaa relaatiotietokannan taulukon riviä, mutta erona on se, että dokumentti on itsenäinen. Sen ei tarvitse sisältää samanmuotoista tietoa kuin muut dokumentit, ja dokumentti voi myös sisältää toisia dokumentteja, eli niillä voi esittää hierarkkisia rakenteita helposti.




```
SELECT "Yhteystapa", "Yhteystieto" FROM "Yhteystiedot"
JOIN "Henkilöt" ON "Henkilöt"."Tunniste" = "Henkilön tunniste"
JOIN "Yhteystavat" ON "Yhteystavat"."Tunniste" = "Yhteystavan tunniste"
WHERE "Sukunimi" = 'Ruuskanen' AND "Etunimi" = 'Seija';
```

Yhteystapa	Yhteystieto
Puhelin (oma)	040 123 4567
Twitter	@ruusseija

Esimerkki 5. SQL-kysely, jolla haetaan tietyn henkilön yhteystiedot.

```
SELECT "Yhteystieto" FROM "Yhteystiedot"
JOIN "Yhteystavat" ON "Yhteystavat"."Tunniste" = "Yhteystavan tunniste"
WHERE "Yhteystapa" = 'Sähköposti';
```

Yhteystieto
markus@igi.fi
timo@suomi99.fi

Esimerkki 6. SQL-kysely kaikkien sähköpostiosoitteiden hakemiseen.

Syvemmälle tekniikkaan

Tietotekniikan ravintoketjussa tietokannat sijoittuvat sovellusohjelmien alapuolelle, eli ne ovat matalan tason tekniikkaa, jonka kanssa käyttäjä ei ole suoraan tekemisissä. Pienet tietokannat sijaitsevat yksittäisen sovellusohjelman omissa tiedostoissa, kun taas suuret ovat palvelinkeskuksissa, joihin sovellusohjelma pitää yhteyttä tietoverkon välityksellä.

Jos haluaa perehtyä tietokantojen tekniikkaan tarkemmin, voi asiaa lähestyä kahta eri tietä. Helpomman polun tarjoavat graafiset sovellusohjelmat kuten Microsoft Access ja Libreoffice Base. Niiden taustalla toimii SQL-kieltä käyttävä relaatiotietokanta, mutta taulukoiden käsittely ja niiden väliset suhteet hoidetaan visuaalisen käyttöliittymän avulla. Ainakin kodin ja pienen yrityksen tarpeisiin nämä ohjelmat riittävät mukavasti.

Ohjelmoijalle lienee mielekkäintä sukeltaa heti kättelyssä syvemmälle ja asentaa jokin erillinen tietokantajärjestelmä sekä opiskella kannan ohjaamiseen käytettävää kyselykieltä. Suosittuja avoimen lähdekoodin relaatiotietokantoja ovat PostgreSQL, MySQL ja Sqlite. Dokumenttipohjaisia järjestelmiä edustavat esimerkiksi MongoDB ja CouchDB. Muunkinlaisia tietokantajärjestelmiä on olemassa.

Tai ehkä sopiva ensimmäinen askel on se, että tutkii jonkin internetissä olevan julkisen tietokannan hakutoimintoja. Samalla voi miettiä, millaisia tietokantarakenteita taustalla voisi olla.

```
{
    "Tunniste": 2,
    "Sukunimi": "Alaluusua",
    "Etunimi": "Markus",
    "Puhelin": {
        "Oma": "050 111 2222",
        "Työ": "016 999 8888"
    }
}
```

Esimerkki 7. Dokumenttipohjainen tietokanta ei perustu taulukoihin vaan itsenäisiin rakenteellisiin dokumentteihin ja metatietoon.

Kiintolevy Commodore 64:ään

Kun Commodore 64 oli uusi, kiintolevyt olivat järjettömän kalliita ylellisyysvarusteita, joista pelejään kasetilta lataavat peruskäyttäjät eivät edes tienneet mitään. Mutta aika kuluu, tekniikka kehittyy, ja nykyisin on aivan perusteltua sovittaa kiintolevyohjain myös vanhaan sotaratsuun.

Teksti: Jarkko Lehti Kuvat: Manu Pärssinen, Jarkko Lehti

Joku voisi kysyä, miksi ihmeessä C64 tarvitsee kiintolevyä. Sille on olemassa hyvä levyasema-emulaattori ja liuta huonompia, joilla ohjelmien lataaminen onnistuu suhteellisen vaivattomasti. Kaikkien levyasemien ja levyasema-emulaattorien yhteinen ongelma on kuitenkin lataamisen nopeus: muistin täyttäminen kestää suhteellisen pitkän ajan jopa levyturbolla. Turbojen yhteensopivuus monessa osassa ladattavien ohjelmistojen kanssa on vähintäänkin huono. Esimerkiksi suosittu kernal-laajennus JiffyDOS on suhteellisen yhteensopiva ohjelmien kanssa, mutta kiintolevyllä varustettu IDE64-laajennus peittoaa sen nopeudessa yli kymmenkertaisesti.

IDE64 on saatavilla suoraan tekijöiltä noin 90–100 euron hintaan hieman mallista riippuen. Saatavuutta voi kysellä englanniksi sähköpostiosoitteesta josef@ide64.org.

Ei vain levyturbo

Latausnopeudet eivät ole IDE64:n ainoa etu. Se sisältää myös aivan uudenlaisen käyttöliittymän tiedostojen hallintaan, loistavan konekielimonitorin, laajan tuen eri levyasemille sekä PCLinkin. PCLinkin kautta voidaan muodostaa yhteys PC-tietokoneeseen, jolloin C64:llä voi ladata, kopioida ja käynnistää ohjelmia suoraan PC:n kiintolevyltä. PCLink osaa hyödyntää C64:n ja PC:n välisessä tiedonsiirrossa

monia erilaisia kaapeliratkaisuja, mutta helpoin lienee IDE64 v.4.1:n integroitu USB-liitäntä.

IDE64:n käyttöohjekirja on myös aivan omaa luokkaansa ja peittoaa 100–0 monet kaupallisetkin ohjekirjat kattavuudessa ja selkeydessä. Jos englanti on hallussa edes jotenkin, ohjekirjaan kannattaa ehdottomasti tutustua, sillä se kertoo laitteen käyttämisestä kaiken ja hieman enemmänkin.

Kompakti salama

Kuten nimestä voi päätellä, IDE64:ssä voi käyttää IDE-liitäntäisiä kiintolevyjä. Järkevintä on kuitenkin käyttää Compact Flash -muistikorttia, joka kuluttaa vähemmän virtaa eikä ole altis kolhuille. Käytännössä kaikki CF-kortit nimittäin tukevat IDE-komentoja. Uusimmissa IDE64-versiossa onkin jo valmiiksi integroituna liitäntä CF-kortille. Vanhemman IDE64:n omistajat voivat käyttää IDE-CF-adaptereita, joita on saatavilla hyvinkin edullisesti esimerkiksi verkkohuutokaupoista.

IDE64 käyttää CFS-tiedostojärjestelmää (Commodore File System). CFS on rakennettu mahdollisimman yhteensopivaksi. Esimerkiksi FAT-tiedostojärjestelmästä poiketen siinä voi käyttää lähes kaikkia PETSCII-merkistön merkkejä tiedostojen nimissä. Myös eri tiedostotyypit ovat mukana, jotta yhteensopivuus alkuperäisen C64-ohjelmiston kanssa on mahdollisimman hyvä.

Alusta alkaen

Vaikka aluksi on varmasti suuri kiusaus tehdä vain yksi iso osio käyttöön, kannattaa niitä kuitenkin tehdä useampi pieni. C64 tikittää vain noin yhden megahertsin kellotaajuudella, ja isolle osiolle mahtuu loputtomasti tiedostoja. Joka kerta kun uusia tiedostoja luodaan, C64 joutuu käymään levyn sisällysluettelon läpi ja merkkailemaan sinne asioita. Mitä enemmän tiedostoja osiolla on, sitä hitaammaksi uusien tiedostojen luominen käy. Hyvä tapa on tehdä esimerkiksi peleille, demoille ja projekteille omat osionsa. Kannattaa esimerkiksi miettiä, kopioiko yli 45 000 tiedostoa sisältävän HVSC-musiikkikokoelman omalle osiolleen vai jonkun muun osaksi.

Kiintolevyn alustus IDE64-käyttöön voi olla asiaan vihkiytymättömälle ylitsepääsemätön prosessi, joten käyn sen tässä läpi kohta kohdalta. PC-koneen käyttöjärjestelmänä on Windows. Esimerkkikorttina käytän Transcendin kahdeksan gigatavun korttia, joka toimii 133×-nopeudella. Kahdeksan gigaa on järkyttävän paljon tilaa C64-ohjelmille, mutta sen kanssa pääsemme paremmin tutustumaan kortin osioinnin ihmeelliseen maailmaan.

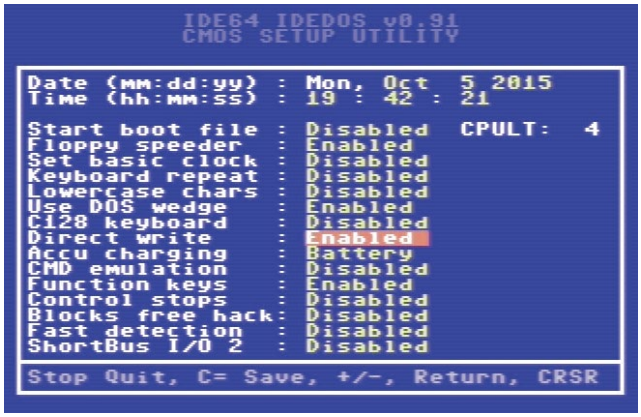
Aivan aluksi pitää C64:lle saada kopioitua osiointiohjelmisto CFSfdisk. Tämä onnistuu helpoiten PCLinkillä. Esimerkissämme käytetään IDE64 v.4.1:n PCLinkkiä USB-yhteydellä.

Ensimmäiseksi Windowsiin pitää asentaa FTDI-piirisarjan ajurit, jotka saa esimerkiksi osoitteesta www.ftdichip.com/Drivers/D2XX.htm.

Kun ajurit on asennettu, voit kytkeä USB-kaapelin IDE64:n portista PC:lle ja kytkeä C64:n virrat päälle. Windowsin pitäisi nyt tunnistaa uusi laite ja asentaa järjestelmään uusi sarjaliikenneportti. Lataa linkkiosiossa mainittu PCLink-ohjelmisto ja pura se johonkin kansioon. Käynnistä kansiosta tiedosto ideservd.exe.

Paina C64:n näppäin ← pohjaan ja napauta RESTORE-näppäintä, jolloin IDE64:n asetusruutu aukeaa.

Valitse STANDARD SETUP ja paina RETURN. Valitse listalta DIRECT WRITE ja vaihda sen tilaksi Enabled painamalla RETURN.



Tallenna tämän jälkeen asetusten muutos painamalla C=-näppäintä ja valitse listasta SAVE & EXIT.

Kirjoita C64:n BASIC-kehotteeseen MAN ja paina RETURN. IDE64:n tiedostonhallinta käynnistyy. Paina C= ja 4. Tiedostohallinta vaihtaa asemaan ”14”, joka on tässä tapauksessa PCLink. Tällöin PC:n kiintolevyt näkyvät listattuna. Valitse asema, johon latasit CFSfdisk-ohjelmiston, ja käynnistä se painamalla tiedoston päällä RETURN.

```
CFSfdisk version 10b
Copyright (C) 2001-2009 Kajtar Zsolt (Soci/Singular)

CFSfdisk comes with ABSOLUTELY NO WARRANTY; This is free
software, and you are welcome to redistribute it under
certain conditions; see LICENSE for details. (GPL-2)

Drive number (4-30): 12

Model number: ATA-CFA KAJTAR ZSOLT (SOCI/SINGULAR)
Serial number: 2015-02-17 23:41:45
Firmware revision: 29352

For LBA disks only the LBA sectors matters.
The default parameters should be fine.

Cylinders (1-65535, default 15538):
Heads (1-16, default 16):
Sectors (1-255, default 63):
LBA available (Y/N, default Y):
LBA (256-268435455, default 15662304):

Using CFS partition entry. (2-15662304)

Command (m for help):
```

CFSfdisk käynnistyy ja kysyy aseman numeroa. IDE64:n CF-kiintolevyn laitenumero on aina oletuksena 12, joten kirjoita `12` ja paina RETURN.

Seuraavaksi CFSfdisk kyselee kiintolevyn perusparametreja. Ne voi ohittaa oletusasetuksilla, joten painele RETURN aina, kun näytölle ilmestyy Command-kehote.

```
Cylinders (1-65535, default 15538):
Heads (1-16, default 16):
Sectors (1-255, default 63):
LBA available (Y/N, default Y):
LBA (256-268435455, default 15662304):

Using CFS partition entry. (2-15662304)

Command (m for help): n
Partition number (1-16): 1
Start (3-15662303, default 3):
Use 123, +123, +123G, +123M or +123K
End (9-15662303, default 15662303): +1G
Partition name: Boot

Command (m for help): p

Drive 12: 15662304 sectors (7647 MiB)
Disklabel: ""

Nr Flags   Start       End Size(KiB) Id System  Name
 1*F--         3   2097154   1048576  1 CFS     Boot

Command (m for help): _
```

Luo uusi osio painamalla `N` ja RETURN. Valitse osion numeroksi `1` ja paina taas RETURN. Start-arvoksi voi hyväksyä oletusarvon, eli paine vain RETURN. Osion kooksi halutaan yksi gigatavu, joten kirjoita `+1G` ja paina RETURN. Osion nimeksi voi kirjoittaa jotain sopivaa, esimerkissä se on `Boot`. Perään taas RETURN.

Tämän jälkeen ohjelma palaa taas komentokehotteeseen. Komennolla `P` voi listata kortille tehdyt osiot. Esimerkissä ei luoda enempää osioita, mutta samalla perusidealla niitä voi luoda kortille useampiakin, kunnes kortti täyttyy. Osiointiin voi palata myöhemminkin, siispä kaikkia osioita ei ole pakko tehdä tässä samalla.

```
Partition name: Boot

Command (m for help): p

Drive 12: 15662304 sectors (7647 MiB)
Disklabel: ""

Nr Flags   Start       End Size(KiB) Id System  Name
 1*F--         3   2097154   1048576  1 CFS     Boot

Command (m for help): w

Drive 12: 15662304 sectors (7647 MiB)
Disklabel: ""

Nr Flags   Start       End Size(KiB) Id System  Name
 1*F--         3   2097154   1048576  1 CFS     Boot

Write this partition table (Y to write): y
Formatting partition 1...done.


*** TURN OFF THE COMPUTER WHEN FINISHED PARTITIONING ***

Command (m for help): _
```

Kirjoita lopuksi tehdyt muutokset kortille komennolla `W`. Vahvista osiotaulun kirjoittaminen näppäilemällä `Y` ja RETURN. Tämän jälkeen osio alustetaan, mikä kestää jonkin aikaa. Jos teit kerralla koko kahdeksangigaisen kortin edestä osioita, tässä voi kestää hyvinkin pitkään. Ei auta kuitenkaan hermostua vaan tehdä jotain muuta odotellessa – ohjelmisto hoitaa hommansa varsin luotettavasti.

Sammuta tämän jälkeen C64:stä virrat ja käynnistä se uudelleen. Nyt tiedostojen tallentamiseen on käytössä yhden gigatavun osio!

Huomaa, että tässä oletettiin kortin olevan tyhjä. Jos se on ollut jo aiemmin käytössä, pitää kortin osiotaulu tyhjentää ennen näiden ohjeiden soveltamista. Tämä onnistuu CFSfdiskin komennolla `O`.

Liitännäisistä lisäiloa

IDE64:n hienouksia on myös tuki erilaisille liitännäisille (plugin), joilla tiedostonhallintaan saadaan avattua esimerkiksi d64- ja t64-tiedostot. Tämä mahdollistaa sen, että yksittäisiä tiedostoja voi kopioida helposti levynkuvista kiintolevylle ja toisinpäin. Levynkuvat voi myös kirjoittaa vaivattomasti lerpulle. SID-musiikin kuuntelu on myös vaivatonta liitännäisen ansiosta: vie vain kursori .sid-tiedoston

Valmis projekti.

päälle ja paina RETURN. Liitännäisten ainoa ongelma on niiden suhteellisen hankala käyttöönotto. Tähän on onneksi tehty valmis paketti.

Liitännäisten asennus valmiista paketista käy hyvin yksinkertaisesti. Lataa paketti netistä ja pura se sopivaan kansioon. Käynnistä PC:llä PCLink-ohjelmisto ja käynnistä C64:n tiedostonhallinta MAN-komennolla. Siirry tiedostonhallinnan alempaan paneeliin painamalla CTRL-näppäintä. Paina C= ja 4, jolloin siirrytään PC:n kiintolevylle. Siirry hakemistoon, johon kopioit plugins-paketin sisällön, ja valitse tiedostot painamalla INS/DEL-näppäintä. Valittu rivi muuttuu keltaiseksi. Kun kaikki kolme riviä ovat valittuina, paina 5, jolloin tiedostot kopioituvat IDE64:n kiintolevylle.

Kun liitännäiset on kopioitu kiintolevyn juureen, sammuta C64 ja käynnistä se uudelleen. Sen jälkeen IDE64-ympäristö on käytettävissä lähes kaikkine herkkuineen.

Käynnistä tiedostonhallinta ja vie osoitin readme-tiedoston päälle. Paina 3, ja tiedoston pitäisi aueta tekstitiedostoja avaavaan liitännäiseen.

IDE64:n laiteohjelmiston päivitys

IDE64:n kehityksessä on kaksi haaraa: vakaaksi merkattu 0.90x-sarja ja huomattavasti kehittyneempi 0.91x. Itse olen käyttänyt jo vuosia 0.91x-sarjan versioita, ja ne tarjoavatkin huomattavasti lisäjoustavuutta liitännäisiin. IDE64:n laiteohjelmiston päivitys on suhteellisen suoraviivainen toimenpide mutta voi tuntua vaikealta, kun sen tekee ensimmäisen kerran. Lienee siis parasta käydä se tässä kohta kohdalta läpi.

Avaa nettiselaimessa seuraava osoite: singularcrew.hu/idedos/beta/compile.php

Ensimmäisessä kohdassa valitaan ympäristö. Tässä on viisainta valita ensimmäinen vaihtoehto: C64/C128.

Toisessa kohdassa valitaan IDE64-laajennuksen versio. Versionumero on painettu piirikortille, josta sen voi tarkistaa.

Kolmannessa kohdassa valitaan haluttu levyturbojärjestelmä. Tässä lienee turvallisinta valita JiffyDOS.

Neljännessä kohdassa valitaan käytetty PCLink-järjestelmä. Tässä kohtaa kannattaa valita oikein, sillä toimivan PCLinkin voi rikkoa valitsemalla tässä väärin. IDE64 V4.1:n kanssa valitsemme tietenkin USB ja Base-osoitteeksi $DE5D.

Viidennessä kohdassa ei tarvitse valita mitään, jos koneessasi ei ole kahta äänipiiriä eri IO-osoitteissa tai et aio käyttää CD-ROM-asemaa IDE64:n kanssa.

Lopuksi paina ”Download beta PEROM” -valintaa.

Pura ladattu idedos-paketti sopivaan kansioon ja käy purkamassa sieltä myös idedos_util-paketti. Käynnistä PCLink-järjestelmä ja siirry IDE64:n tiedostonhallinnalla PC:n kiintolevylle ja siihen kansioon, johon purit idedos_util-paketin. Kopioi ”perom 1.9f.prg” kiintolevylle painamalla näppäintä 5. Siirry sitten takaisin kansioon, josta varsinainen .rom-päätteinen tiedosto löytyy. Nimeä se selvyyden vuoksi uudelleen muotoon ”ide.rom” painamalla tiedoston päällä näppäintä 6. Vaihda tiedostonhallinnan paneelia painamalla CTRL ja lataa tiedosto ”perom 1.9f”. Painamalla SHIFT + RETURN tietokone palaa BASICiin. Älä kuitenkaan käynnistä ladattua ohjelmaa.

Oikosulje IDE64:n ohjelmointijumpperi ja resetoi tietokone IDE64:n takana olevasta reset-painikkeesta. Tietokone käynnistyy normaalisti BASICiin. Huomaa, että BASICin vapaana näyttämä RAM-muistin määrä on hieman pienempi kuin ilman kytkettyä IDE64-moduulia. Poista ohjelmointijumpperi.

Kirjoita SYS2059, jolloin PEROM Programmer -ohjelmisto käynnistyy. Valitse ”UPDATE OVER USB PCLINK”, jolloin ohjelma kysyy tiedoston nimeä. Muokkaa riville nimi ”IDE,ROM”. Huomaa, että tiedostopäätteen erottimena on pilkku eikä piste. Paina lopuksi RETURN, jolloin laiteohjelmiston päivitys käynnistyy.

Kun ruudulla lukee ”UPDATE SUCCESSFUL”, tietokoneen voi käynnistää uudelleen, jolloin päivitetty laiteohjelmisto tulee käyttöön.

Apulerpulla alkuun

Vaihtoehtoisesti laiteohjelmiston voi päivittää myös lerpulta. Kopioi silloin Perom Programmer ja ide.rom lerpulle ja lataa ohjelmistot ohjelmointijumpperin sulkemisen jälkeen normaalisti sieltä. Huomaa myös Perom Programmer -ohjelmassa oleva optio tallentaa nykyisestä ohjelmistosta varmuuskopio lerpulle. Näin voit palata aiempaan versioon helposti, jos ongelmia ilmenee. Vaihtoehtoisesti voit käyttää tämän artikkelin laadinnan yhteydessä syntynyttä idetoolit.d64-levykuvatiedostoa, jonka saat osoitteesta skrolli.fi/2015.4. Saat kirjoitettua sen lerpulle kopioimalla sen PC:n kiintolevylle ja painamalla RETURN sen päällä IDE64:n tiedostonhallinnassa.

Laiteohjelmiston päivityksen jälkeen on syytä huomata, että PCLink-asema on yleensä laite 13 eikä 14, kuten tässä artikkelissa on kerrottu.

IDE64 sisältää myös diagnostiikkatilan, jonka saa aktivoitua käynnistämällä C64:n SHIFT/LOCK-näppäin pohjassa. Diagnostiikka käy läpi RAM- ja ROM-muistin ja ilmoittaa mahdollisista ongelmista ruudulla. Konetta käynnistettäessä tulevan mustan ruudun, joka kertoo käynnistyksen aikaisista asioista, saa pysymään näytöllä pitämällä käynnistyksen aikana CTRL-näppäintä pohjassa.

Tässä oli vasta pintaraapaisu IDE64:n monipuoliseen maailmaan. Muistakaa lukea laitteen ohjekirja, sillä se on oikeasti vielä parempi kuin artikkelin alussa kehuin.

Linkkejä

- CFSfdisk: singularcrew.hu/ide64warez/site/Tools/System/CFSfdisk10b.zip
- PCLink-ohjelmisto: singularcrew.hu/idedos/beta/ideserv-0.29c-20150810.zip
- FTDI-ajurit: www.ftdichip.com/Drivers/D2XX.htm
- IDE64:n käyttöopas: singularcrew.hu/idedos/beta/IDE64_users_guide_beta.pdf
- IDEDos beta rom: singularcrew.hu/idedos/beta/compile.php
- CFS-tiedostojärjestelmä: singularcrew.hu/idedos/cfs.html
- IDE64-liitännäispaketti: ide64.org/plugins2012.zip
- IDE64:n kotisivu: ide64.org/
- IDE64-uutisia: news.ide64.org/
- IDE64-pelejä ja -ohjelmia: singularcrew.hu/ide64warez/
- Latausnopeusmittauksia: singularcrew.hu/idedos/perf.php
- Artikkelin oheistiedosto: skrolli.fi/2015.4

NYKY-BASICIT
Perustason ohjelmointikieli tänä päivänä

Basic on hyvin tunnettu, yksinkertainen, vuorovaikutteinen ja monissa eri ympäristöissä tuettu kieli. Esittelyssä muutamia tarjolla olevia kaupallisia ja ei-kaupallisia basic-järjestelmiä.

Teksti: Antti Ylikoski Kuvat: Mikko Torvinen, Mitol Meerna, Antti Ylikoski

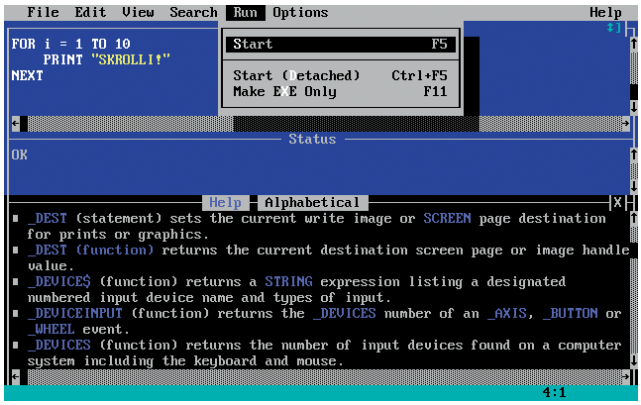
Ei-kaupalliset basicit

FreeBASIC

FreeBASIC on GPL-lisensoitu, komentorivipohjainen kääntäjä Windowsille, DOSille ja Linuxille. Luultavasti sen saisi suhteellisen pienin vaivoin toimimaan myös Applen OS X:llä. Freebasicissa on sen oletustoimintamoodin lisäksi QB-moodi, jossa se on yhteensopiva Microsoftin Quick Basicin ja QBasicin kanssa. Ohjelma käännetään QB-moodissa esimerkiksi seuraavalla käskyllä:

```
C:\BASIC>fbc -lang qb QBOHJELMA.BAS
```

Siirryin itse FreeBASICista QB64:ään löydettyäni FB:stä selvän toimintavirheen. Käyttämäni ohjelma ei toiminut FreeBASICilla, koska taulukoiden määrittely DIM-käskyllä toimi näissä järjestelmissä eri tavoin. Kokemusteni perusteella voin kuitenkin suositella FreeBASICia ohjelmoinnin opetteluun ja harjoitteluun mutta en mihinkään, missä kaiken täytyy toimia 110-prosenttisesti.



QB64

QB64

QB64:n on tarkoitus olla nykykäyttöjärjestelmillä toimiva, lähdekoodiyhteensopiva korvike Microsoftin QBasicille. QB64:ään kuuluu ruutueditori ja yksinkertainen integroitu kehitysympäristö, jotka muistuttavat hyvin läheisesti QBasicin vastaavia. Toki myös ulkopuolista tekstieditoria voi käyttää.

Olen päätynyt valitsemaan ei-kaupallisten basicien joukosta omaksi työkalukseni QB64:n. Käytössäni on joitakin keskikokoisia, muutaman sadan tai tuhannen rivin Basic-ohjelmia, joista QB64 on tähän mennessä suoriutunut moitteettomasti. Ainoatakaan virhettä ei järjestelmästä ole löytynyt.

QB64:n nettisivuilla on tarjolla niin Wikimedia-muotoinen dokumentaatio, esimerkkiohjelmia kuin keskustelufoorumitkin. Suosittelen.

OpenOffice BASIC

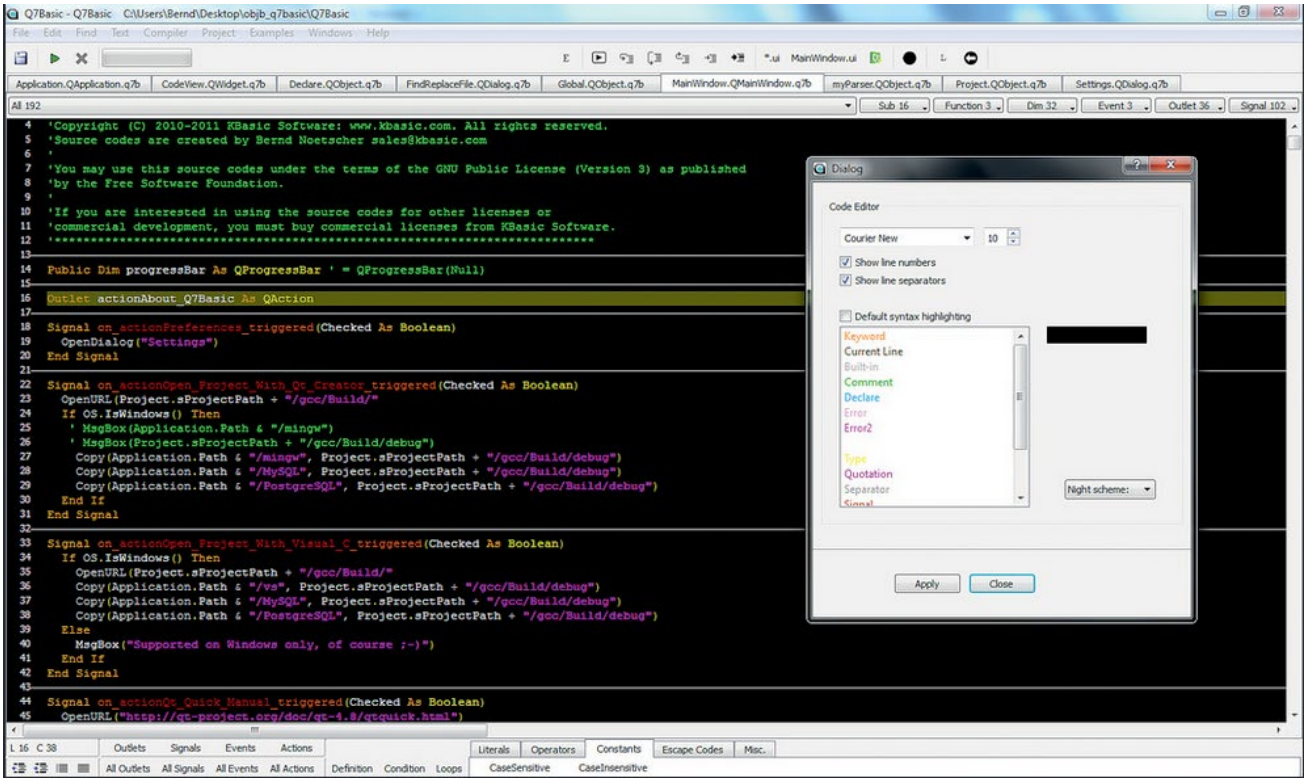
OpenOffice BASIC löytyy tavallisesta Apache OpenOfficesta makrojen hallinnan alta. Se eroaa jonkin verran Microsoft-tyyppisistä basiceista, ja sen erot niihin on erikseen opeteltava. Muut tämän artikkelin freeware-basicit noudattavat varsin läheisesti de facto -standardia eli QBasicia ja Quick Basicia.

Kun yritin käyttää OpenOffice BASICia yksinkertaisten kokeiluohjelmien tekemiseen, se kaatuili alinomaa. En näin ollen voi suositella OpenOffice BASICia oikean työn tekemiseen. Pikku ohjelmiin se käy, kunhan opiskelee sen de facto -standardista poikkeavat piirteet ja saa sen toimimaan kaatuilematta.

Kaupallinen puoli

Q7Basic

Q7Basic, Basic for Qt tai aikaisemmalta nimeltään Kbasic on huokea, Qt-käyttöliittymäkirjastoon pohjautuva basic-tulkki. Sen työkalut graafisten käyttöliittymien toteuttamiseen ovat



Q7Basic

melko kehittyneet. Haittapuolena on tulkkipohjaisuuden tuoma hitaus, joka koituu ongelmaksi esimerkiksi pitkää ja raskasta laskentaa suoritettaessa.

Q7Basic on kuitenkin ohjelmointiteknisesti sangen hyvä ja rinnastettavissa täysihintaisiin kaupallisiin basic-järjestelmiin tukensa ja ominaisuuksiensa puolesta.

PowerBASIC

PowerBASIC on melko hyvin maailmalla tunnettu kaupallinen basic-järjestelmä. Siihen saa joukon erilaisia komponentteja, kääntäjän, IDE:n ja lukuisia aputyökaluja. Se on hyvin QBasic-yhteensopiva vaikka sisältääkin runsaasti laajennuksia siihen verrattuna. PowerBASICin kehittäjien ylpeyden aiheena on erään suuren kansainvälisen pankin järjestelmä, joka koostuu 3 miljoonasta rivistä PowerBASIC-koodia.

Visual BASIC

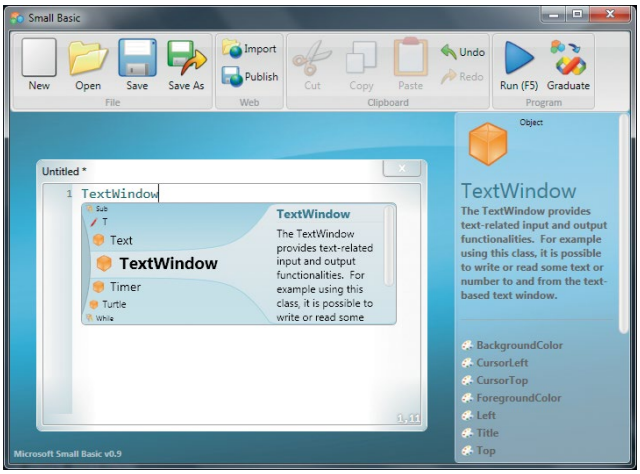
Itse asiassa ”Visual BASIC” tarkoittaa kahta eri järjestelmää. Alkuperäinen Visual Basic toimitettiin Visual Studio 6.0:n ja sitä aiempien versioiden mukana, kun taas uudempi Visual BASIC .NET kuuluu Microsoftin .NET-alustan varustukseen. Nämä basicit eivät ole keskenään yhteensopivia. Lisäksi on vielä olemassa klassisen Visual BASICin kanssa yhteensopiva VBA (Visual BASIC for Applications), joka kuuluu Microsoftin Office-pakettiin.

Visual BASIC on hinnoiteltu kaupallisena tuotteena. Se on ohjemistoteknisesti korkeatasoinen, tukee hyvin Microsoft-ympäristöä ja sen integroitua kehitysympäristöä on pidetty sangen hyvänä. Näistä syistä VB on mielestäni suositeltavin basic kaupallisiin, teollisiin, tieteellisiin ja erityisesti sotilassovelluksiin.

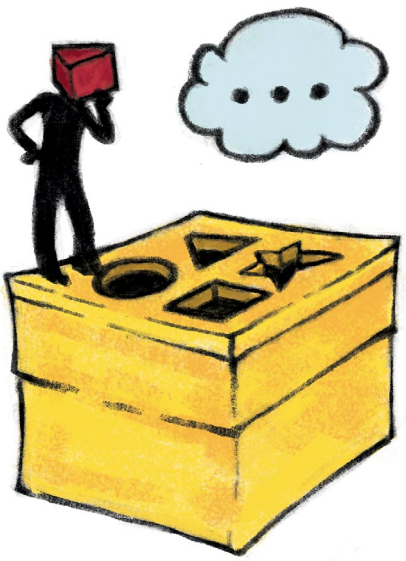
Visual BASIC 6 -ohjelmat toimivat edelleen käytännössä kaikissa Windows-järjestelmissä ilman ylimääräisiä korjauksia. Visual BASIC on käytettävissä myös Windows RT -mobiilikäyttöjärjestelmässä, jonka pariin Microsoft on pyrkinyt paimentamaan ohjelmistokehittäjiä.

Microsoft Small BASIC

Microsoft Small BASIC on Visual BASICin yksinkertaistettu versio, joka on tarkoitettu etenkin lapsille ja ohjelmoinnin aloittelijoille. Sen yhteydessä tulee mukana koko joukko harjoituksia ja esimerkkejä. Yhteensopivuutensa vuoksi Small BASICia voi hyvin käyttää ohjelmoinnin harjoitteluun ennen Visual Basiciin siirtymistä.



Microsoft Small BASIC



Kuningaskunnan kalkulaattori – ZX Spectrum Vega

Brittiläinen Sinclair-yhtiö teki 80-luvun alkupuolella sarjan huikean menestyneitä kotimikroja. Niistä rakastetuin oli ZX Spectrum, joka on kotimaassaan Isossa-Britanniassa samassa asemassa kuin Commodore 64 Suomessa. Nyt Spectrum on tehnyt paluun retromikrorintaman etulinjaan.

Teksti: Jukka O. Kauppinen

Kuvat: Retro Computers, Jukka O. Kauppinen, Bill Bertram / Wikimedia Commons

Kiittäkäämme taas kerran joukkorahoituksen jumalia, sillä nämä ovat olleet suosiollisia monelle menneiden vuosikymmenten rakkaalle teokselle ja monille uusillekin luomuksille. Pelaajatkin ovat saaneet nauttia hyvistä peleistä ja erikoisista pelikoneista. Myös Spectrum-mikron paluu tulevaisuuteen onnistui joukkovoimalla, tällä kertaa Indiegogo-palvelussa järjestetyn keräyksen kautta.

Spectrumin paluuseen toivottiin 100 000 punnan pottia, ja rahaa kertyikin kiitettävät 156 000 puntaa. Sillä pystyttiin tekemään klassinen kasarimikro uusiksi nykytekniikalla. Mutta miten paluu vuoteen 1982 onnistuu?

Emulaattorimikroillaan

ZX Spectrum Vega on nykytyyliin emulaattorimikro. Se ei sisällä ensimmäistäkään piiriä tai molekyyliä alkuperäisestä vuoden 1982 laitteesta, vaan se on ARM-prosessorin ympärille rakennettu ”System on Chip”, tekijöidensä mukaan täydellinen Spectrum-järjestelmä. Sen voi uskoa, sillä Vegan taustalla on todella raskaan sarjan tekniikkaporukkaa, joka ymmärtää laitteen sielunelämän. Sovitaan siis, ettei se ole pelkkä emulaattori, vaan vähän enemmän. Sen huomaa kyllä peleistäkin, sillä ne pyörivät Vegassa tavanomaista emulaattoria paremmin.

Laite on myös ideoitu aivan uudella tavalla. Alkuperäisen kotimikron toisinnon sijaan Vega on kämmenelle mahtuva tv-peli, jossa peliohjain on integroitu itse koneeseen. Ulkoasuun on silti löydetty vähän Spectrum-henkeä.

Mutta kuten Spectrumissakin, myös Vegassa on asioita tehty yksinkertaistaen ja säästäen. Esimerkiksi näppäimet ovat alkuperäistä paremmat, mutta niitä on niin tautisen vähän. Spectrum oli sentään täysiverinen tietokone, jossa oli näppäimistö. Vegassa sen sijaan on eräänlainen peliohjaimen korvike, jossain on ristikko-ohjain ja kahdeksan näppäintä. Näillä kyllä pystyy pelaamaan vaivattomasti suurinta osaa Spectrumin peleistä, mutta silti – miksei tähän voi liittää edes ulkoista USB-näppäimistöä?

Mainittakoon toki se historiallinen yksityiskohta, että suurella osalla spectrumisteja ei luultavasti ollut joystickiä. Mikro oli äärimmilleen niukistettu, ja jopa joystick-liitin oli lisävaruste. Niinpä kaikkia Spectrum-pelejä piti pystyä pelaamaan myös näppäimistöllä. Ja kun suurimmalle osalle peleistä riitti nelisuuntainen ohjaus ja fire-nappi, niin mikäs tässä.

Ja parempihan tämä versio on kuin se aivan alkuperäinen prototyyppi, jossa oli vain ristiohjain ja neljä nappia. Kahdeksalla napilla onnistuu vähän monimutkaisempienkin pelien hallinta.

Mutta miten pelit sitten on oikein sorvattu Vegan ihmenamiskoille? Jo-

ZX Spectrum oli halpa rimpula – ja juuri siksi yksi 1980-luvun suosituimmista kotimikroista. Kuva: Bill Bertram / Wikimedia Commons.

ZX Spectrum

Julkaisuvuosi: 1982
Prosessori: Zilog Z80A, 3,5 MHz
Muisti: 16 tai 48 Kt RAM, 16 Kt ROM
Näyttö: 256×192 pikseliä, 8 väriä
Äänipiiri: yksikanavainen sisäänrakennettu kaiutin

Lue lisää Spectrumista ja muista 1980-luvun kotimikroista Skrollin numerosta 2014.2: Tasavallan tietokoneen haastajat.

kainen laitteen mukana tuleva peli on sovitettu yksilöllisesti Vegalle niin, että tärkeimmät toiminnot ovat aina suurin piirtein samat. Yksinpeli alkaa tuosta, tämä on tulitusnamiska ja niin edelleen. Toimintojen logiikkaa on mietitty, ja vaikka sitä ei muistaisikaan, niin näppäimet näytetään aina pelin alussa, ja apuruudun saa esille napinpainalluksella. Julmetun hieno ominaisuus on pelitilan dumppaaminen kokonaisuudessaan muistiin haluttaessa, eli minkä tahansa pelin voi näin tallentaa milloin tahansa. Nyt ei tarvitse nauhoitella tallennusta kasetille ja kirjoittaa ylös kentän lopussa saatavaa koodia!

Käyttäminen on muutenkin verrattain selkeää. Vegassa ei ole edes virtakytkintä. Se vain tökätään mihin tahansa USB-virtalähteeseen ja liitetään televisioon komposiittipiuhalla. Kyllä, komposiitilla. HDMI olisi toki ollut tervetullut, mutta eivätpä alkuperäisen Spectrumin pelit tämän kummoisempaa tarvitse. Myönnetään, että kuvanlaatu on silti erinomainen ja Vegan saa näin useampiin telkkareihin ja näyttölaitteisiin kiinni.

Outo pelivalikoima

Yksi Vegan tärkeistä moraalisista valttikorteista on tuhannen pelin valikoima, joista suuri osa on lisensoitu oikeuksien haltijoilta. Prosessissa on tiemmä aukkoja, mutta sanotaan näin: Vegaan on lisensoitu enemmän pelejä kuin millekään muulle Spectrum-emulaattorille. Kyllä tätä pitää jo vähän arvostaa.

Mutta ne pelit, ne pelit. Tuhat peliä, joista iso osa on aivan kamalaa roskaa. No, sitähän ne 1980-luvun alun Spectrum-pelit suurelta osalta olivat. Mutta vaikka selaan pelitarjontaa sen kummemmin Spectrum-historiaa muistelematta, niin silti törmään aukkoihin. Moni klassinen ja tärkeä peli puuttuu. Lisenssiongelmia, olettaisin. Ymmärrettävää mutta raivostuttavan kiusallista, koska tällaisenaan Vegan tarjonta antaa ZX Spectrumin pelihistoriasta liiankin todenmukaisen kuvan. Ei Commodore 64:kään näyttäisi kovin hyvältä, jos käteen lyötäisiin tuhat satunnaista peliä.

Niinpä tarjolla on nyt valtava määrä amatöörikyhäelmiä ja outopelejä, joista kukaan ei ole koskaan kuullutkaan. Helmien ja klassikoiden löytäminen tai tunnistaminen vaatii pelisilmää.

Vielä kurjempaa on seikkailupelien valtava määrä. Enkä sano tätä siksi, että inhoaisin niitä. Mutta kuvitelkaapa itse millaista on pelata tekstiseikkailua, siis semmoista peliä johon komennot kirjoitetaan näppäimistöllä sanoina ja lauseina, ilman näppäimistöä. Niin. ILMAN NÄPPÄIMISTÖÄ, virtuaalinäppäimistöllä. Se on pyllystä, täysin mahdotonta.

Ja tämä on hanurista. 80-luvun klassikkotietokoneen mukana tulee satoja pelejä, joita ei pysty pelaamaan ollenkaan. Kyllä on pohjamutia ruopattu, että saadaan tuhat peliä täyteen.

Onneksi pelivalikoiman puutteisiin löytyy ratkaisu. Vegassa on näet paikka muistitikulle, jolla voi sekä päivittää vehkeen firmiksen että käyttää muita emulaattorimuodossa olevia pelejä ja ohjelmia. Niiden näppäinkartan sorvaukseen joutuu käyttämään hetken aikaa, mutta saapahan näin uusio-Spectruminsa peli- ja ohjelmakirjaston mieleisekseen.

Niin kiva, mutta kallis

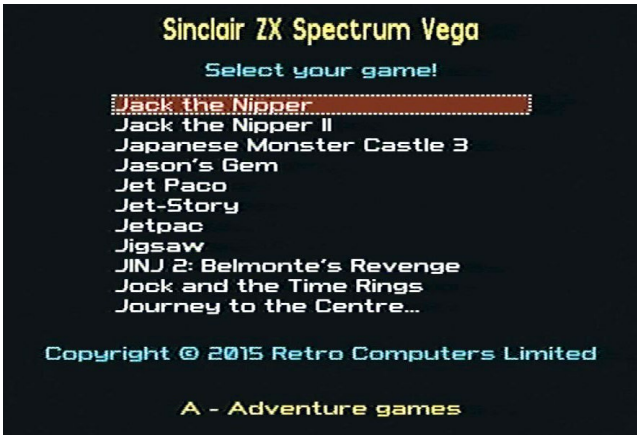
ZX Spectrum Vega on tekniseltä kannalta katsoen erinomainen brittihittimikron ylösnousemus. Laite on pieni ja elegantti, joskin myös tv-pelimäisyydessään lelumainen. Se mahtuu kouraan, napit on sijoitettu hyvin ja pelaaminen on mukavaa. Yksinkertaistettu ja pennyjä säästävä rakenne on tavallaan jopa perinteitä kunnioittavaa, mutta voi itku ja parku: miksei koneessa ole USB-liitintä ulkoiselle näppäimistölle?

Kriittinen mikroilija, jolla ei ole tunnesiteitä Spectrumiin, tosin yskähtelee taatusti hintalapun kohdalla. Joukkorahoitetun Vegan sai noin satasella, myynti-Vega maksaa Briteistä tilattuna 150 euroa ja postikulut päälle. Se on iso raha vieraan vallan hittimikrosta. Jos Spectrum olikin kuningaskunnan tietokone, niin meillä se jäi alkukisan jälkeen hopealle tai pronssille. Siihen nähden Vegan iloihin on syytä suhtautua terveellä epäilyksellä.

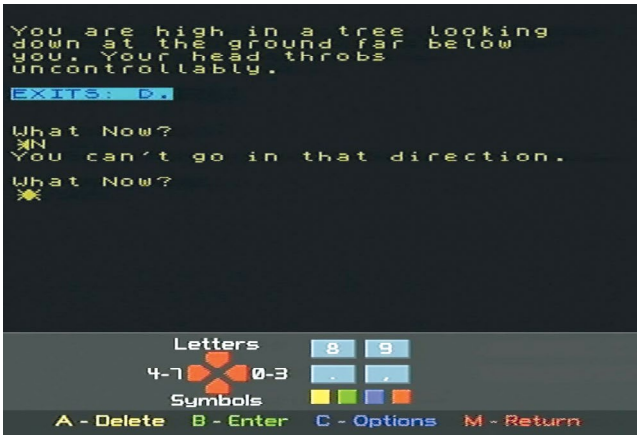
Minulle henkilökohtaisesti Vega on kuitenkin hintansa arvoinen. ZX Spectrum tarjosi minulle Jussin luona pelikokemuksia ennen C64:n saapumista raukoille rajoillemme, ja muistelen monia sen pelejä lämmöllä. Uskallan myös tunnustaa, että jotkut pelit olivat Spectrumilla parempia. Olihan se hassu rimpulakone, mutta kyllä se oli silti yksi niistä suurista kulmakivistä, joille eurooppalainen ja suomalainen mikroilu- ja pelikulttuuri rakentuivat.

Joten pelataan lisää. Kunhan ensin löydetään seuraava hyvä peli. Se tässä on vaikeinta.

ZX Spectrum Vega
Valmistaja: Retro Computers
Myyjä: Funstock
Lisätietoja: www.funstockretro.co.uk/zx-spectrum-vega-console



Vegan pelit on lajiteltu toiminta- ja seikkailupeleihin. Aakkosellisissa valikoissa liikutaan nuolinäppäimillä.



Vegan suurin puute: virtuaalinäppäimistöllä ei pelata tekstipelejä.

ZX Evolution – ihmekone itänaapurista

Neuvostoajan niukasti tunnettu kotitietokonekulttuuri oli yhtäältä vientirajoitusten ja moninaisten muiden saatavuusongelmien värittämää, mutta toisaalta yhtä innostunutta kuin lännessäkin. Jos C-64 oli Suomessa tasavallan tietokone, niin ZX Spectrum klooneineen valloitti yhtä lailla brittien kuin neuvostomikroilijoiden sydämet.

Teksti ja kuvat: Markku Reunanen

Neuvostoliiton tietokonekulttuuri erosi merkittävästi läntisestä: Commodoreja, Atareita ynnä muita täkäläisiä suosikkeja ei maassa juuri nähty, mutta esimerkiksi Yamaha vei maahan MSX-tietokoneita. Lisäksi valtion siunaamien yritysten, kerhojen ja harrastajien voimin suunniteltiin ja rakennettiin monenkirjavia laitteita. Itäblokin kotimikroilla oli usein tunnetut läntiset esikuvat, ja kloonit ajoivatkin niiden ohjelmia joko suoraan tai pienten muutosten jälkeen.

Пентагон ja sen aikalaiset

Yleisimpiin kotitietokoneisiin lukeutuivat Sinclair ZX Spectrum -kloonit, joita tunnetaan kymmeniä eri malleja. Ulkonäkö saattaa tosin hämätä, sillä sama piirilevy voi löytyä hyvinkin eri näköisten kuorien sisältä. Näppäimistöjä ja koteloita improvisoitiin niistä osista, joita sattui olemaan saatavilla. Varhaiset Spectrum-kloonit vastaavat ominaisuuksiltaan 48 kilotavun malleja, mutta myöhempiin malleihin lisäiltiin myöhemmistä länsikoneistakin tuttuja laajennuksia levyasemasta lisämuistiin ja PSG-äänipiiriin.

Spectrumin sydän on useaa eri toimintoa hoitava Ferrantin ULA-piiri, jonka kloonaamisen sijasta sama logiikka toteutettiin erillisillä, yksinkertaisemmilla piireillä. Muita venäläisten koneiden luonteenomaisia piirteitä ovat normaalin laajennusväylän puuttuminen, runsas DIN-liitinten käyttö, RGB-ulostulo sekä joystick-liitäntä, tietenkin DIN-liitännällä.

Levyasema tuli viralliseen Spectrum-perheeseen vasta lopun aikoina, +3:n myötä, ja jäi hieman kuriositeetiksi. Venäjällä Beta Disk -kloonit olivat edistyneemmissä koneissa vakiotavaraa. Sekä Beta Disk että sen komentotulkki TR-DOS ovat nekin alkujaan Englannista peräisin.

Rakentelijoiden ja demoskenen piirissä suurimman suosion saavutti 1990-luvun taitteessa ilmestynyt Pentagon (Пентагон), joita on niitäkin useampaa eri mallia. Vakiokokoonpanossa – jos sellaista on edes olemassa – on ainakin Beta Disk, stereo-PSG sekä 128 kilotavua muistia. Suorittimen ei tarvitse kilpailla muistiväylästä näytönvirkistyksen kanssa, joten kone on myös hieman nopeampi kuin aito Spectrum.

Suurin osa brittiläisistä ohjelmista toimii Pentagonilla joko suoraan tai niistä on valmiit modifioidut versiot, mutta toiseen suuntaan tilanne onkin toinen: levytiedostoja ei voi soittaa kasettiporttiin, minkä lisäksi venäläisten demojen käyttämät ajastukset aiheuttavat länsikoneilla monenlaista toimimattomuutta.

Neuvostoliiton romahtamisen jälkeen venäläiset harrastajat omaksuivat nopeasti uusia tekniikoita myös Spectrumien rakenteluun: tee se itse -asenne ja syvällinen elektroniikan tuntemus heillä oli jo valmiiksi. Käsin kasaan juotetut piirilevyt johtovyyhteineen väistyivät kauniisti suunniteltujen ja ammattimaisesti valmistettujen emolevyjen tieltä. Erilliset piirit katosivat ohjelmoitavan FPGA:n sisuksiin, liittimet vaihtuivat loputtomista DINeistä PC-standardin mukaisiksi, muistimäärä nousi megatavuihin, suorittimiin

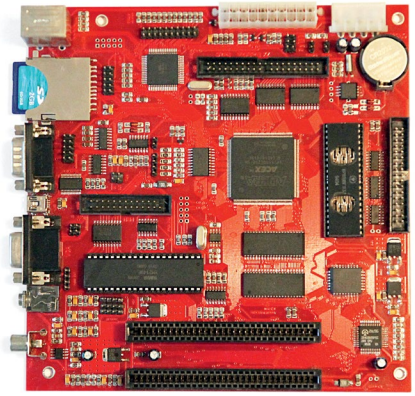
Valikkokäyttöinen Spectrum? Kyllä Venäjällä osataan!

tuli turbomoodeja, ja monet aiemmin erilliset laajennukset integroitiin osaksi vakiokonetta. Monet näistä edistyneistä malleistakin kantoivat edelleen perinteikkäästi Pentagonin nimeä, vaikka niillä ei enää ollutkaan sanottavasti tekemistä alkuperäisten laitteiden kanssa.

Evoluution seuraava askel

Pentagon-sarjan uusimpia tulokkaita edustaa ZX Evolution, joka tunnetaan toisinaan myös nimellä ”Pentevo” tai lyhyemmin ”Evo”. NedoPC-ryhmän jäsenten kehittämä mini-ITX-kokoinen emolevy on moderniin tapaan FPGA-pohjainen ja sisältää yhdessä paketissa paljon toiminnallisuutta. Levyasema on korvattu SD-kortinlukijalla, muistia on peräti neljä megatavua, laitteesta saa ulos VGA-signaalin, ja hiiren sekä näppäimistön saa kiinni tavallisella PS/2-liittimellä.

Z80-suoritinta voi ajaa paitsi 3,5 megahertsillä, myös 7 ja 14 megahertsin turbomoodeissa. Eräänä tärkeänä piirteenä ZX Evolution noudattaa näytönvirkistyksessä ja muistinkäsittelyssä samoja ajastuksia kuin Pentagon.


Evolutionin saa joko rakennussarjana tai valmiiksi koottuna emolevynä.

Edistyneelle harrastajalle on tarjolla monia virittely- ja laajennusmahdollisuuksia: emolevyllä on liitännät korppuasemalle sekä IDE-levyille, ja erääseen piikkirimoista voi kytkeä alkuperäisen Spectrumin näppäimistön sekä Atari-joystickin. Ohjelmistojen osalta ZX Evolution tarjoaa halukkaille kurkistusikkunan TR-DOS:n omalaatuiseen maailmaan, joka jäi useimmilta kokematta 1980-luvulla.

Evoa saa ainoastaan Venäjältä tilaamalla, ja sen saatavuuskin tuntuu vaihtelevan tekijöiden mielenkiinnosta riippuen. Itse onnistuin moisen hankkimaan, eikä hinnaksi tullut kuin reilu sata euroa kaikkine kustannuksineen. Pelkällä piirilevyllä ei tee yksinään mitään, joten lisäksi tarvitaan vielä näppäimistö, muistikortti sekä PC-kotelo virtalähteineen.

Näyttöliitäntä on tavallinen VGA, mutta kaikki näytöt eivät laitteen tuottamasta signaalista pidä. Itse ratkaisin ongelman vetämällä emolevyn RGB-piikkirimasta kaapelin SCART-liittimeen, jonka kautta televisiot ja videomonitorit suostuvat näyttämään asianmukaista 50 hertsin kuvaa.

Käynnistyttyään ZX Evolution menee vakiona ”Evo Reset Service” -päävalikkoon, josta voi säätää laitteen perusasetuksia, ladata SCL- ja TRD-levytiedostoja virtuaalisiin asemiin tai hypätä eri BASIC-versioihin. Valikkoa voi käyttää myös hiirellä, vaikkei siitä merkittävää hyötyä olekaan. Alkuperäisen Spectrumin pelit myytiin lähes yksinomaan kaseteilla, ja niinpä Evokin osaa ladata emulaattorien käyttämiä TAP-tiedostoja suoraan muistikortilta. Hieman autenttisempaa kokemusta etsivä voi halutessaan päristää ääntä suoraan takanurkasta löytyvään kasettiliittimeen.

Pelit eivät ole erityisen herkkiä ajastuksille, ja kaikki TAP-tiedostoista kokeeksi lataamani pelit näyttivät toimivan kuten pitääkin. Pelkkään pelaamiseen ei kuitenkaan ZX Evolutionia tarvita, joten entäpä ne demot? Niitä voi toki nykyään ajaa emulaattorilla, jolloin jää kuitenkin paitsi aidosta 50 hertsin päivitystaajuudesta sekä tiukasta kuvan ja äänen synkronoinnista. Testasin noin kolmeakymmentä tunnetuinta venäläistä demoa, eikä yhdenkään kanssa ilmennyt ongelmia. Jokunen pelkille länsikoneille tarkoitettu tuotos puolestaan sekosi ajastuksissaan, ja efektit muuttuivat sotkuksi.

Virittelijän paras товарищ

FPGA-piiri on täysin mahdollista ohjelmoida uudestaan, mikä mahdollistaa eräänlaiset rautapäivitykset. Evolutioniin tulee toisinaan firmwarepäivityksiä, joissa on korjattu ongelmia tai lisätty uusia ominaisuuksia. Päivittäminen saattaa kylläkin nostaa kielitaidottomalle kylmän hien pintaan, sillä lähes kaikki ohjeet ovat venäjäksi. Dramaattisemmin laitteen toimintaa muuttaa TS-Confin nimellä tunnettu erillinen firmware, joka tuo toimintatilojen kirjoon vielä yhden vaihtoehdon lisää. Omaa tiedostomuotoaan käyttävä TS-Conf sisältää muun muassa parannettuja 2D-kiihdytettyjä grafiikkatiloja, jolloin ollaan jo varsin kaukana Sir Cliven alkuperäisistä laitteista.

Vaikkei ZX Evolutionissa olekaan 1980-luvun nostalgista hohdetta, on se Spectrum-harrastajalle mainio hankinta. Retrokoneiden viimeaikaisen hinnannousun myötä sen hinta on jo suorastaan edullinen, etenkin kun huomioi, kuinka paljon toiminnallisuutta mukana seuraa.

Oman kokoelmani länsikoneet ovatkin pysyneet visusti kaapin hyllyllä Evon hankinnan jälkeen. Teknisiä vikoja ei ole parin vuoden aikana vastaan tullut – ainoa merkittävä kompastuskivi on oikeastaan ollut vain kielimuuri, jonka taakse jää pääosa virittelyyn ja ohjelmointiin tarvittavasta dokumentaatiosta.

Lisätietoja: nedopc.com/zxevo/zxevo_eng.php

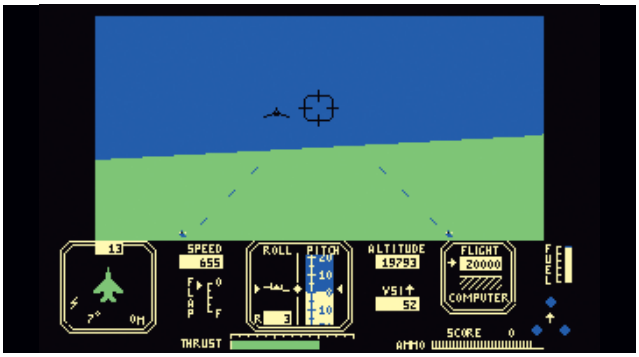
10 peliä ZX Spectrumille

Spectrum oli Iso-Britanniassa Se Virallinen tietokone, jolla oli eniten käyttäjiä, faneja, suurimmat lehdet ja parhaat pelikehittäjät. Moni legendaarinen pelisuunnittelija ja -ohjelmoija aloitti juuri Spectrumilla. Se oli myös kuusnelosta nopeampi muun muassa 3D-grafiikassa. Mitä hyvää Spectrumille sitten tehtiin?

Teksti: Jukka O. Kauppinen

Ant Attack (1983)

Quicksilvan seikkailua voidaan hyvällä syyllä pitää 3D-pelien ja isometristen seikkailujen esi-isänä. Se oli ensimmäinen teos, jossa kuvakulma nostettiin taivaalle ja maailma kuvattiin kolmiulotteisena kokonaisuutena. Tiukalla konekielellä koodattu luomus oli myös todennäköisesti maailman ensimmäinen selviytymiskauhupeli.

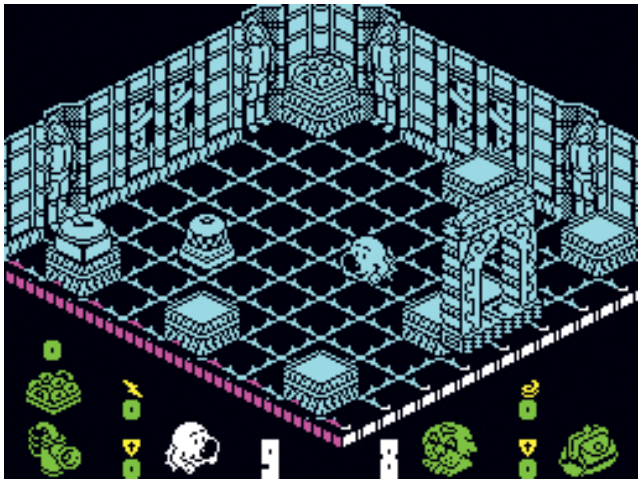


Fighter Pilot (1983)

Ehkä maailman ensimmäinen kunnon lentosimulaattori, jossa riitti nopeutta, toimintaa ja modernia ilmasotaa. F-15 Eagle -hävittäjän mallinnus oli ehkä alkeellista, mutta silti tämä F-15 tykitti lujempaa kuin mikään. Tekijäfirma Digital Integration kasvoikin yhdeksi 80-luvun tärkeimmistä simufirmoista.

Great Escape (1986)

Oceanin The Great Escape haki innoituksensa tositapahtumiin perustuvasta Suuri pakomatka -sotaelokuvasta. Pelaaja on sotavanki, jonka täytyy päästä pakoon saksalaisten sotavankileiriltä. Peli oli aikaisekseen äimistyttävä, sillä avoin pelimaailma pyöri pelaajan ympärillä ja tämän tekemisistä riippumatta. Vangin täytyi osallistua pakollisiin leiritoimintoihin ja yrittää etsiä niiden lomassa pakenemiseen tarvittavia välineitä, kaivaa pakotunnelia ja piilotella vartijoilta. Selkeä viivavektorigrafiikka pyöri Spectrumilla paremmin kuin millään toisella pelikoneella, ja esimerkiksi meille tutumpi C64-versio oli selvästi hitaampi ja kankeampi.

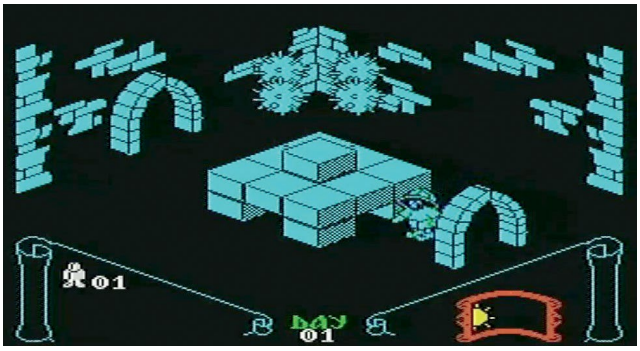


Head Over Heels (1987)

Jon Ritmanin (Batman, Match Day) mestariteos viilasi herran Spectrum-koodausloiston täydellisimmilleen. Edellisvuonna julkaistu isometrinen Batman-seikkailu haastoi jopa Ultima Play The Gamen pelien loistavuuden, ja nyt taitavaan suunnitteluun ja viilattuun pelimoottoriin yhdistyi myös sympaattisen persoonallinen päähenkilö (tai oikeastaan kaksi). Hyvät puzzlet, söpöily ja hauskat hahmot nostivat sen yhdeksi Spectrumin ikivihreimmistä klassikoista.

Jet Set Willy (1984)

Manic Minerin jatko-osa oli suurempi, valtavampi ja massiivisempi kuin mikään peli ikinä. Se myös kumpuaa brittiläisen kasaritasoloikkasuunnittelun syvimmästä ytimestä: staattisilla ruuduilla liikkuu kaikenlaisia öttiäisiä, joita pelaajan hahmon täytyy väistellä yli hyppimällä ja tarkalla ajoituksella. Hipaise jotain ja kuolet. Willyily oli yhtaikaa sekä hirvittävän turhauttavaa että lumoavaa, ja se olikin yksi 80-luvun eniten myydyistä videopeleistä Britanniassa.



Knight Lore (1984)

Jos listaamme pelejä, jotka mullistivat maailmaa, niin Knight Lore on oksalla ylimmällä, yhdessä Wolfenstein 3D:n kanssa. Ultimate Play The Gamen luomus oli teknisesti ja konseptuaalisesti aikaansa edellä, ja sen Filmation-pelimoottori käytännössä loi isometristen 3D-seikkailujen genren.

Knight löytyykin käänteentekevyytensä ansiosta maailmaa eniten mullistaneiden videopelien top 10 –listalta, ja se innoitti monia aikansa pelinkehittäjiä, ei vähiten John Ritmania, jonka Batman-pelistä voit lukea Skrolli 2014.1:stä. Pelissä seikkaili Ultimaten luottosankari Sabreman.

Laser Squad (1988)

Julian Gollopin luoma Laser Squad loi pohjan muun muassa Advance Wars -pelisarjalle ja oli suora edeltäjä Gollopin UFO/X-COM-megahiteille PC:llä. Huima science fiction -sotapeli oli vuoropohjainen taktinen strategia, jossa kaksi joukkuetta soti erilaisissa ympäristöissä. Äärimmäisen taitavasta pelisuunnittelusta syntyi teos, jota pelataan aktiivisesti edelleen.

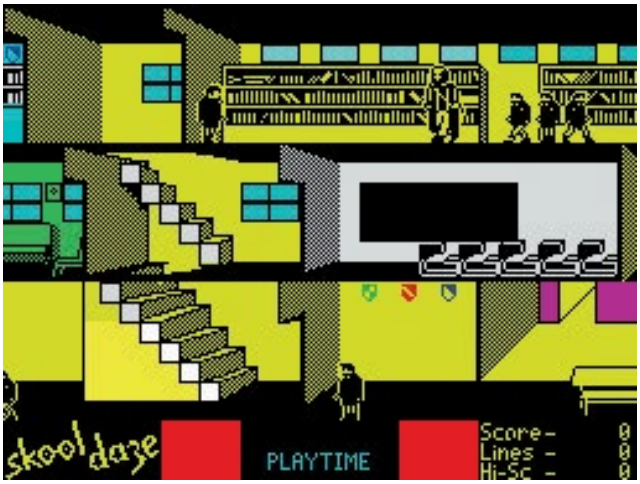
Julian Gollopin on muuten yhä aktiivinen pelialalla. Pitkän Ubisoft-kauden jälkeen herra julkaisi äskettäin PC:lle fantasiastrategia Chaos Rebornin. Alkuperäinen Chaos ilmestyi – tietenkin – ZX Spectrumille.

Manic Miner (1983)

Matther Smithin mestariteos muutti tasohyppelyt ikiajoiksi. Sen raivostuttavuus ja koukuttavuus löysivät vertaisensa vasta Smithin myöhemmästä Jet Set Willystä. Vaikka Manic Miner oli pohjimmiltaan huima peli, oli se myös niin helvetillisen vaikea, ettei monikaan taatusti päässyt ikinä viitostasoa pidemmälle. Jos etsimme pikselintarkkojen tasoloikintojen esi-isää, niin tätä kauemmas ei tarvitse katsoa.

Sabre Wulf (1984)

Ultimate Play The Gamen varhaisiin hitteihin lukeutunut Sabre Wulf oli vauhdikas, värikäs ja hauska sokkeloseikkailu, jossa pelaajat tutustuivat ensimmäisen kerran Utimaten useissa peleissä seikkailleeseen Sabremaniin. Teos jäi sankareineen kasibittisen pelaamisen historiaan ja sementoi Tim ja Chris Stamperin aseman pelaamisen legendoina. Herrojen seuraava pelitalo muuten tunnetaan nimellä Rare.



Skooldaze (1984) / Back to Skool (1985)

Skooldaze on yhä edelleen ideana äimistyttävä. Ai peli, jossa ollaan koulussa? Jep. Ainutlaatuinen idea hiekkalaatikkomaailmatoteutuksella ja itsenäisesti toimivilla tekoälyn ohjaamilla sivuhahmoilla pohjusti huumorilla höystettyä seikkailua, jolla riittää lumovoimaa vaikka muille jakaa. Peli viehätti sekä avoimella puuhastelulla että ilkikurisella asenteellaan. Sitä jatko-osineen pidetään muun muassa The Sims -pelisarjan esi-isänä.

Sillä kyllähän peli, jossa voi ampua opettajaa ritsalla takaraivoon ja töniä luokkatoverit pulpetin äärestä, jos joka paikka on varattu, on ihan täysimittainen koulusimulaatio mutta vain koulunkäynnin parhaimpien puolien kera.

BK-0011: Rautaa rautaesiripun takaa

Neuvostotietotekniikkaa ei ole kovin helppo haalia käsiinsä. Yritin jo kymmenisen vuotta sitten hankkia itselleni Venäjältä БК- eli BK-kotitietokonetta, mutta toimitusketju kohtasi epäonnea. Jari ”Jaffa” Jaanto onnistui kuitenkin äskettäin huutamaan Ebaysta yhden tällaisen koneen ja lainasi sen minulle tutkittavaksi.

Teksti: Ville-Matias Heikkilä

Kuvat: Manu Pärssinen, Matti Hämäläinen, Ville-Matias Heikkilä

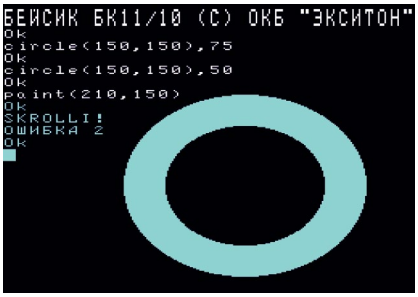
BK:ta voidaan pitää ainoana aidosti neuvostoliittolaisena kotitietokoneena. Muut neukkumikrot, kuten Radio-86RK, Vektor-06 ja erilaiset Spectrum-kloonit, saivat useimmiten alkunsa harrasteprojekteina ja perustuivat länsimaista kopioituihin suorittimiin. BK sen sijaan syntyi osana virallista ohjelmaa ja oli mikropiirejä myöten täysin kotoperäinen.

Koneesta ja sen komponenteista vastasi Tieteellinen Keskus (Nautšnyj Tsentr), joka oli Neuvostoliiton elektroniikkaministeriön alaisuudessa toiminut tieteellis-teollinen yhtymä. Sen toiminta keskittyi Moskovan lähellä sijaitsevaan Zelenogradin kaupunkiin, jota on pidetty Neuvostoliiton vastineena Kalifornian Piilaaksolle. Kaupunki oli neuvostoaikaan salainen ja suljettu samaan tapaan kuin vaikkapa avaruustekniikan ja kosmonautiikan keskus Tähtikaupunki.

BK:n suoritin on nimeltään K1801VM1, ja se suorittaa PDP-11-käskykantaa. Suoritinta ei kuitenkaan alkujaan suunniteltu tälle käskykannalle vaan yhteensopivaksi Tieteellisen Keskuksen oman NC-1-minitietokoneen kanssa. 1980-luvun vaihteessa itätietotekniikka oli kuitenkin jäämässä jälkeen länsimaista, joten elektroniikkaministeriö halusi kiriä eroa umpeen sitoutumalla entistä vahvemmin ”kansainvälisiin standardeihin”. Niinpä K1801:lle laadittiin uusi mikrokoodi, joka teki siitä PDP-11-yhteensopivan. Suoritin päätyi sittemmin hyvin monenlaisiin laitteisiin satelliiteista taskutietokoneisiin.

BK:n edeltäjinä voidaan pitää Tieteellisen Keskuksen aikaisempia PDP-11-yhteensopivia henkilökohtaisia tietokoneita, joita ovat esimerkiksi DVK-sarja ja Tetriksen alkuperäisalustana tunnettu Elektronika 60. BK suunniteltiin kuitenkin nimenomaan edulliseksi koti- ja koulumikroksi länsimaisen mallin mukaan: sen ensisijaisena massamuistina toimii kasettinauhuri, ja 32 kilotavun RAM-muistista peräti puolet on varattu 512×256 mustavalkopikselin grafiikkanäytölle. Tarjolla tosin on myös muistinsäästömoodi, jossa näyttö on vain muutaman rivin korkuinen. Näppäimistönä alkuperäisessä vuoden 1985 BK-0010:ssa on tunnoton muovilevy, joka tosin myöhemmissä malleissa vaihtui kirjoituskonetyyppiseksi.

BK purettiin laatikostaan Simulaatio-demotapahtumassa.



Eksiton-suunnittelutoimiston Basic muistuttaa syntaksiltaan Microsoftin tulkkeja.

Muistimonitori. Muistipaikkaan 77776 on kirjoitettu PDP-11:n kuuluisa itseäänkopioiva käsky MOV -(PC),-(PC) ja käynnistetty se.

Kone ja liittimet

Haltuuni saamani kone on BK-0011, jonka tuotanto alkoi vuonna 1989. Sen tärkein uudistus on RAM-muistin laajennus 128 kilotavuun. 0011 on siitä harvinainen välimalli, että se ei ole täysin yhteensopiva 0010:n kanssa. Yhteensopivuus nimittäin palautettiin jo tuotannon alkuvaiheessa, jolloin uudeksi mallinimeksi tuli 0011M. Monet 0011M-harrastajat laajensivat koneitaan edelleen muun muassa äänipiireillä sekä levyke- ja kiintolevyasemilla, joita esimerkiksi koneelle tehdyt demot hyödyntävät järjestään. Tällaisia luksusvarusteita ei Ebay-paketista kuitenkaan paljastunut – sisällä oli vain koneen keskusyksikkö, virtalähde ja pari johtoa.

Mihin virtalähde kytketään? Koneen takana on kahdenlaisia liitäntöjä: kaksi pitkää suorakaidemaista väyläliitäntää ja neljä viisipinnistä DIN-liitäntää. Laitteita tutkittuani päädyin siihen, että käyttöjännite otetaan kirjaimilla БП merkitystä kolosta, sillä muuntajamötikän pohjassa ovat sanat ”Блок Питания” eli ”virtalähde”. Muuntaja rupesi käynnistettäessä sirisemään ilkeästi, mikä ei valanut luottamusta maadoittamattoman, metallikuorisen itätuotteen käyttöturvallisuuteen. Kone kuitenkin piippaili lupaavasti näppäimiä painettaessa.

BK:n videoulostulosta saa kahdenlaista signaalia: mukana tullut videokaapeli antaa televisioon kytkettynä mustavalkoista 512×256 pikselin kuvaa, mutta tarjolla on myös RGB-pinnit värikuvaa varten. Värikuvan resoluutio on 256×256 – kutakin mustavalkokuvan pikseliparia vastaa yksi leveämpi väripikseli. Värivalikoima on insinööripalettien insinööripaletti: musta, punainen, vihreä ja sininen. 0010-käyttäjät joutuivat tyytymään näihin väreihin, mutta 0011:ssä on tarjolla kaikkiaan 16 eri palettia. Pelit ja demot on suunniteltu värigrafiikalle, kun taas muut ohjelmat suosivat isompaa resoluutiota. BK-käyttäjillä lieneekin ollut käytössä kytkimiä, joilla mustavalko- ja värikuvan välillä voi vaihtaa vaivattomasti.

Näppäimistön vasemmalla puolella on luukku, jonka takaa paljastuu kaksi piirikantaa. Ylemmässä kannassa on neuvostovalmisteinen EPROM-piiri, ja alempi on tyhjillään. Käyttäjille on siis tarjolla mahdollisuus vaihtaa koneen ROM-ohjelmistoa varsin helposti. Tyypillisiä ROM-ohjelmistoja ovat Basic- ja Focal-tulkit, muistimonitorit ja levymuisteja hyödyntävät käyttöjärjestelmälataimet. Kannalle asennetulta piiriltä käynnistyi muistimonitori, jonka tekstit oli käännetty englanniksi – kuinka huomaavaista!

Ohjelmaa muistiin oktaalina

Muistimonitori on ominaisuuksiltaan samantapainen kuin konekielimonitori, mutta ilman assembly-esitysmuotoa: muistin sisältöä voi tutkia ja muokata oktaalimuodossa, ohjelmaa voi ajaa halutuista osoitteista normaalisti tai askel kerrallaan, ja dataa voi myös ladata ja tallentaa kasettiliitännän yli. Assembly-esityksen puutteeseen tottui kuitenkin lopulta aika nopeasti, sillä PDP-11-konekieli on alusta asti suunniteltu oktaaliystävälliseksi. Näppäimistön nurkassa huomiota herättävä punainen СТОП-näppäin (”stop”) osoittautui konekielikokeiluissa käteväksi: se saa aikaan keskeytyksen, joka palauttaa suorituksen takaisin muistimonitorin komentoriville.

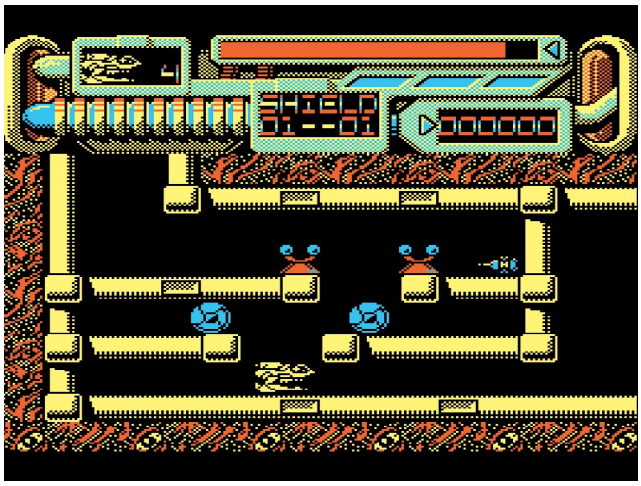
Testailin laitteen toimintaa aluksi pienillä grafiikkaohjelmilla, joilla halusin etenkin kokeilla laitteen suorituskykyä. Näyttömuistia täyttävästä vaakaskrollausrutiiniista tein Youtube-videonkin (”PDP-11 machine code on a BK-0011”). Edistyneempiä kokeiluja en kuitenkaan jaksanut tehdä oktaalikoodia naputtamalla, vaan sitä varten koneeseen pitäisi saada ladattua ohjelmia ulkoisesta lähteestä.

BK:n kasettiporttiin sopivan DIN-liittimen saa nyysittyä esimerkiksi AT-aikakauden PC-näppäimistöstä, ja massamuistilaitetta matkimaan voi valjastaa vaikkapa Arduinon. En ehtinyt kuitenkaan saada omaa viritystäni toimimaan ennen lehden valmistumista, sillä 0011:n kasettitallennusformaatti on epäyhteensopiva muiden BK:iden kanssa, ja liitäntöjen nastakohtaisia kuvauksia oli todella vaikea löytää. Olen siis tyytynyt monimutkaisempien ohjelmien rakentelussa toistaiseksi emulaattoriin.

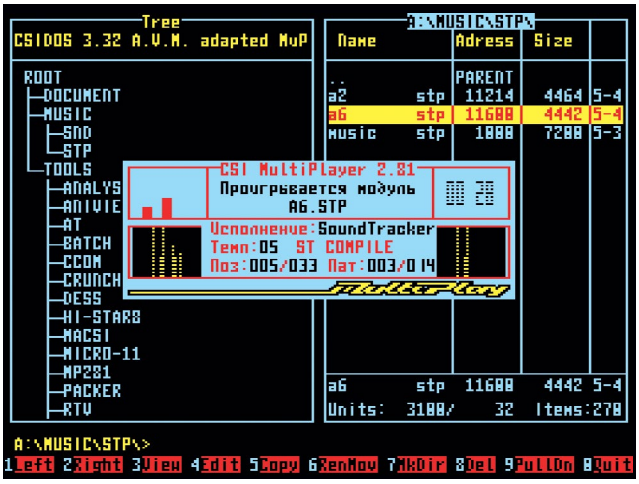
Demoako tekemään?

Mitä tällaisella koneella tekisi? Jo kymmenen vuotta sitten suunnitelmissani oli tehdä BK:lle ensimmäinen länsidemo, eikä kukaan näytä vieläkään ehtineen ensin. Projektia varten pitää kuitenkin ensin syventyä laitteiston heikkouksiin ja vahvuuksiin.

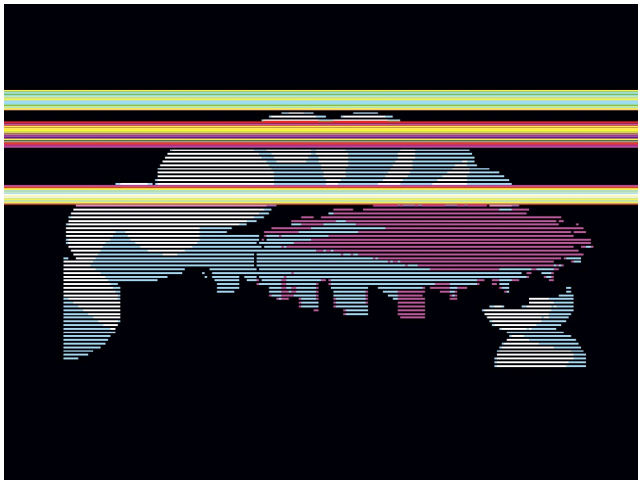
BK on aidosti 16-bittinen laite muistiväyliään myöten, mutta se on omi-



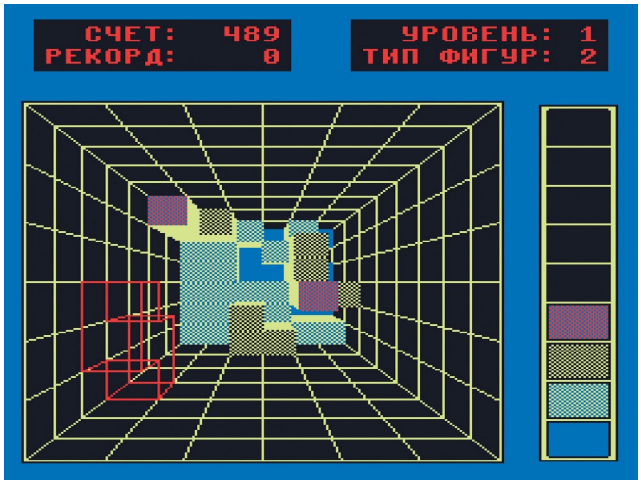
CSI:n Cybernoid-käännös on parhaita BK:n länsipelisovituksia.



CSIDOSin tiedostonhallinta ja AY-piiriä käyttävä musiikkisoitin.



Rasteripalkkeja Mega Best Softwaren demosta Technological Dream.



Kahdensuuntaisen kulttuurivaihdon tulos: Blockoutin BK-käännös.

naisuuksiltaan hyvin ”kasibittinen”. Suorittimen kellotaajuus on mallista riippuen kolmen ja neljän megahertsin välillä, mutta käskyt vievät tuhottomasti kellojaksoja – muistinkäsittelykäskyt järjestään vähintään 20. Alkuperäinen neukkukäskykanta ehkä sai ytimestä enemmän irti, mutta tällaisena suoritin ei ole sen nopeampi kuin vaikkapa Spectrumissa käytetty Z80. Edes kerto- ja jakolaskukäskyjen varaan ei voi toivoaan laskea, ne kun puuttuvat kokonaan. Monimutkaisia muistiosoituksia kannattaa sen sijaan hyödyntää, sillä se on kautta linjan nopeampaa kuin saman tehtävän toteutus yksinkertaisemmilla käskyillä.

BK:ssa ei ole muita näyttötiloja pikseligrafiikan lisäksi, eikä suorittimessa riitä potku 16 kilotavun näyttömuistin uudelleenkirjoittamiseen joka näytönvirkistyksellä. Demoissa ja peleissä tyydytäänkin usein piirtämään vain joka toiselle juovalle. Grafiikkaraudan ainoa ”kiihdytysominaisuus” on mahdollisuus määritellä näyttömuistin alkuosoite 64 tavun tarkkuudella, mikä mahdollistaa pikselintarkan pystyskrollauksen ilman muistin kopiointia. Rasterikeskeytyksiä saa aikaan ajastinkikkailulla, mutta ne ovat käyttökelpoisia lähinnä värimäärän lisäämiseen ja näytön juovakohtaiseen pimentämiseen. Näytönohjaus on niin yksinkertainen, että mahdollisuudet sen muunlaiseen huijaamiseen ovat liki olemattomat.

Näissä puitteissa BK-demon efekteissä kannattaa painottaa algoritmin älykkyyttä raakaoptimoinnin sijaan. Esimerkiksi koko näytön kokoinen pyörivä kuutio toteutettaisiin parhaiten taulukoimalla värirajojen sijainnit ja päivittämällä vain niitä näyttömuistin kohtia, joissa muutoksia tapahtuu. Monimutkaisemmat pikselinpyörittelyt lienee paras pitää pienessä osassa ruutua ja jättää muu tila staattiselle tai pystyskrollaavalle ympärysgrafiikalle.

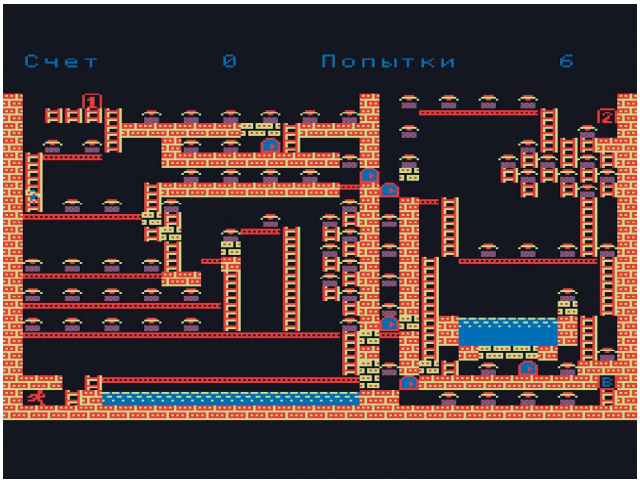
Laajentamattoman BK:n ääniominaisuudet rajoittuvat sisäiseen pietsopiipperiin samaan tapaan kuin alkuperäisessä Spectrumissa. Muutamat pelit yrittävät saada piipperistä enemmän irti alkuruuduissaan, joissa kaiken suoritinajan voi uhrata äänen tuottamiseen, mutta demokäytössä kaipaa kyvykkäämpää äänentuotantotapaa. 128 kilon Spectrumeista tuttua AY-3-8910-äänipiiriä käyttävä laajennus yleistyi BK-käyttäjien keskuudessa 90-luvulla, samoin 8-bittisen PCM-äänen mahdollistava Covox.

BK:n kilpailijat

Jotta BK:n ominaisuudet saisi oikeisiin mittasuhteisiin, on sitä verrattava muihin neuvostoharrastajan ulottuvilla olleisiin kotimikroihin. Teknisesti ehkä varteenotettavin BK:n kilpailija oli kahden moldovalaisinsinöörin vuonna 1987 julkistama Вектор 06Ц eli Vektor 06C.

Vektor päihittää perus-BK:n helposti niin grafiikassa kuin äänessäkin. Vektorissa ovat tarjolla samat grafiikkaresoluutiot kuin BK:ssakin mutta 4 ja 16 värillä. Värit voidaan valita vapaasti kaikkiaan 256 värin kokonaispaletista. Grafiikkamuisti on lisäksi järjestetty kerroksittain, mikä helpottaa esimer-

"Voittamaton"-avaruuspelissä lisävarusteiden hinnat ovat dollareissa.



Suosittua Lode Runner -kloonia КЛАД ("Klad") julkaistiin monta osaa vuosina 1987-1991.

kiksi peliohjelmointia – monissa peleissä on yksi grafiikkakerros varattu kokonaan softaspriteille. Äänestä vastaa PC:stä tutun ajastinpiirin klooni, joka tarjoaa kolme kanttiaaltokanavaa. Vektorista puuttuu kuitenkin rautaskrollaus, sen 8-bittinen suoritin on hidas, ja muisti rajoittuu 64 kilotavuun. Vektorilla ei myöskään ollut takanaan laajaa valtiokoneiston tukea kuten BK:lla.

Sekä Vektor että BK jäivät ennen pitkää Spectrum-kloonien ja myöhemmin PC:n jalkoihin. Spectrumin suurimpana etuna oli valtava länsipiraattiohjelmiston määrä. Grafiikka on teknisesti rajoitetumpaa kuin BK:n tai Vektorin – 256×192 pikseliä, ja kussakin 8×8 pikselin solussa voi käyttää vain kahta väriä kuudestatoista – mutta sitä on nopeampi piirtää. Näin Spectrumilla oli myös tekninen valtti neukkukoneisiin nähden. Vektorin ja BK:n parhaat ominaisuudet yhdistävä kone olisi kuitenkin pärjännyt Spectrumille paljon paremmin, ja ehkä paikallinen tietokonekulttuuri olisi muotoutunut erilaiseksi, jos tällainen kone olisi kehitetty.

Ohjelmatarjonta

BK-sarjan koneet olivat suosituimpia neukkumikroja, jos länsikoneiden kloonit jätetään huomiotta. Esimerkiksi pelejä löytyy niille netistä huomattavasti enemmän kuin millekään toiselle itäblokin koneelle.

Suuri osa BK:n peleistä on varsin yksinkertaisia harrastajaklooneja länsipeleistä: Lode Runner, Pacman, Boulder Dash. Joukkoon mahtuu myös kunnianhimoisempia tekeleitä – joista niistäkin valtaosa on kopioita Spectrumin ja PC:n peleistä: Cybernoid, Joe Blade, Monkey Island 2, Prince of Persia. Joskus peliin on kopioitu alkuperäisen pelin intrografiikat suoraan, mutta itse peli on vain vähän sinnepäin. Toisinaan taas käännös on hyvinkin uskollinen.

Aidosti omaperäistä pelitarjontaa on varsin vähän, toisin kuin vaikkapa Spectrum-klooneilla. Ehkä BK-käyttäjillä oli tarve keskittyä todistelemaan, että BK:llekin saa länsimaisen kaltaisia ohjelmia, kun taas Spectrum-kloonien käyttäjät saattoivat käyttää energiansa uuden luomiseen ja vaikkapa aina vain hulppeampien Elite-kloonien kehittelyyn. Ainoa vastaani tullut hieman Eliteä muistuttava BK-peli on Непобедимый ("Voittamaton"), joka vaikuttaa lähinnä modernisoidulta versiolta 1970-luvun merkkipohjaisesta Star Trek -pelistä.

Alkuperäinen BK käytti massamuistinaan yksinomaan c-kasettia, joten levykäyttöjärjestelmien suhteen ei ole minkäänlaista yleisesti hyväksyttyä standardia. Standardin syntymättömyyteen vaikutti varmasti myös se, että neuvostoteollisuus keskittyi pelkkiin laitteisiin jättäen ohjelmistopuolen rehottamaan omillaan.

Osa käyttöjärjestelmistä (muun muassa ОС БК-11) on PDP-11:ssä yleisesti käytetyn RT-11:n johdannaisia ja käyttää sen tiedostojärjestelmää, mutta suurin osa hylkää RT-11-perinteen. ANDOS väittää käyttävänsä MS-DOSista tuttua FAT12:ta, mutta todellisuudessa venäläisinsinööri on tulkinnut speksejä sen verran omaperäisesti, ettei tiedostoja saa normaalein keinoin ehjinä ANDOS-levykkeille. Yhteensopimattomuus menee niin pitkälle, että monet ohjelmat on pitänyt sovittaa erikseen kullekin käyttöjärjestelmälle, joiden alta ne ladataan: tästä kertoo ohjelmissa usein näkyvissä oleva "adapted by" -teksti.

Pelien introruudut kertovat länsi- ja itäblokkien tietokonekulttuurien eroista: kräkki-intron sijaan näkyviin saattaa tulla ruutu, jossa pelin kopioinut taho ilmoittaa, minkä pelien kopioita hänellä on myytävänä ja millaisiin hintoihin. Etenkin Neukkulan hajottua oli tietokoneharrastajillakin ilmeisesti halua nyhtää rahaa toisiltaan henkensä pitimiksi. Mutta eipä silti, kyllä BK-ohjelmien alussa "cracked by" -rivejäkin näkyy. Näyttäisi jopa siltä, että kopiosuojaukset olivat usein jonkin välikäden eivätkä pelin alkuperäisen tekijän laittamia.

Venäläiset löysivät länsimaistyyppisen demoskenen kilpailukeskeisine partyineen 1990-luvun puolivälin aikoihin. Tällöin järjestettiin muutamana vuonna myös BKmania – yksinomaan BK:lle omistettu demotapahtuma. Tämän jäljiltä on olemassa jonkinlainen määrä BK-demoja, joskin niiden tuotanto hiipui aika reilusti vuosituhannen vaihduttua. Mainitsemisen arvoisia BK-ryhmiä ovat Mega Best Software sekä CSIDOS-käyttöjärjestelmää ja RockMonitor-musiikkiohjelmaa kehittänyt Caroline Software Inc.

BK ei ehkä ole palkitsevimpia alustoja harrasteohjelmoinnin kannalta, mutta tämä tarkoittaa toisaalta myös sitä, ettei sen kaikkia kykyjä ole vielä ehditty hyödyntää täysipainoisesti. Haasteita näyttäisi siis riittävän länsieurooppalaisellekin demo-ohjelmoijalle vielä yllin kyllin.

Pelejä tuunaamassa, osa 1
Pelieditorien aamunkoitto

Pelaajien luoma pelisisältö, kuten modit, ovat kullan arvoisia. Ne kasvattavat pelien kaupallista ikää ja edesauttavat myyntiä. Ne eivät ole uusi keksintö – videopeleissä on ollut kenttäeditoreita ja sisältötyökaluja jo yli 30 vuoden ajan.

Teksti: Jukka O. Kauppinen

Kuvat: Tapio Lehtimäki, Jukka O. Kauppinen, Mobygames, Wikimedia Commons

Miksi peliin pitäisi tehdä sisältöä itse? Eikö se ole pelintekijöiden tehtävä? Siinä missä joillekin riittää valmiista paketista nauttiminen, haluavat toiset luoda itse. Ja jos taidot eivät riitä ihan kaikkeen, on luontevaa muokata valmista peliä – etenkin jos siihen tarjotaan järkevät työkalut. Kenttiä ja muuta sisältöä tehdään ja tehtiin omaksi iloksi ja kavereiden kera nautittavaksi, ja mikseipä sitä jaettaisi muillekin, jos se on teknisesti mahdollista.

Skrolli sukeltaa editoripelien historiaan ja käsittelee tässä artikkelissa genren historiaa noin vuosituhannen vaihteeseen saakka. Uudemmat tuulet puhaltavat numerossa 2016.1!

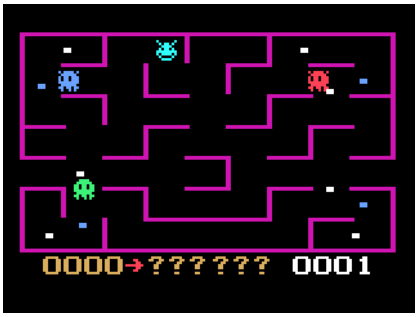
Lode Runnerista se alkoi

Tiettävästi ensimmäinen pelin mukana tullut kenttäeditori löytyi vuonna 1981 julkaistusta Munchkin –konsolipelistä. Pac-Man-kopio nähtiin Suomessakin, ja se oli monelle pikku pelaajalle ensimmäinen kosketus pelisuunnitteluun. Tunnetuimman jälkensä historiaan se tosin jätti Atarin ja Philipsin tekijänoikeuksien historiaa tehneessä oikeudenkäynnissä.

Suuremman loiskauksen aiheutti Lode Runner (1982/1983), jonka kenttäeditori tosin oli enemmänkin osoitus tekijän laiskuudesta kuin innovatiivisuudesta. Arkkitehtuuriopiskelija Douglas E. Smith ohjelmoi pelin alkujaan huvikseen, mutta kun Broderbund hyväksyi pelin julkaistavaksi, oli firmalla kaksi vaatimusta: animaatioista täytyi tehdä paljon paremmat ja kenttiä oli oltava 150.

150, mietti Smith kauhuissaan, ja teki kenttäeditorin jota naapuruston kakarat osasivat käyttää. Hyvä näin, sillä penskat sekä testasivat Smithin tekemiä tasoja että tekivät omiaan. Tiemmä iso osa pelin kentistä onkin teetetty lapsityövoimalla.

Lode Runnerista tuli hitti. Se julkaistiin kaikille aikalaismikroille ja jopa NES-konsolille. Kaikki versiot sisälsivät tuikitärkeän kenttäeditorin, josta otettiin kaikki irti. Esimerkiksi Computer Gaming World -lehti järjesti useita Lode Runner -kentäntekokilpailuja. Näin yhä useampi kuuli pelistä ja kenttien teosta tuli niin suosittua, että jatko-osa, Championship Lode



Runner, koostui yksinomaan pelaajien luomuksista.

Vähän Loden jälkeen saapui samantyyppinen tasoloikinta Wizard (1984), josta julkaistiin myös päivitetty Ultimate Wizard. Myös siinä oli kentänrakennustyökalut, mutta EA:n peli jäi Lode Runner -kuumeen jalkoihin.

Näin pelihistoria sai uuden, tuottavan käänteen. Kenttäeditorit olivat tulleet jäädäkseen, ja kansalaisille oli annettu avaimet pelien muokkaamiseen.

Ensikosketus aiheeseen tuli Philips Videopac G7000 -pelimoduulissa Munchkin, jossa pääsi suunnittelemaan "Pacmanille" uutta labyrinttiä. Olihan se hauska motittaa kummitukset jo heti kättelyssä tai pistää kaverit pelaamaan umpikujaan päätyvää labyrinttiä. Sega SC-3000-kotitietokoneelle tuli aikansa buumipeli, Lode Runner, johon tuli myös tehtyä omia pelitasoja speedruntyyliseen pelailuun. Alkuinnostuksen jälkeen pelkkä ratojen suunnittelu ei tuntunut oikein riittävän, kun halusi suuremmin vaikuttaa pelimaailmaan ja sen tapahtumiin.

Chris Skog

Devaajana säädän editoreissa työnkin puolesta, mutta edelleen suurimpana motivaattorina on sisäinen vimma rakentaa ja luoda jotain, sekä uteliaisuus ja kokeilunhalu. Ensimmäinen kosketus editoreihin oli Philips Videopacin Munchkin, vuosi taisi olla 1983. Munchkin julkaistiin 1981, joten se lienee ensimmäisten editoripelien joukossa.

Santtu Huotilainen

Kasibittistä editointia

Lode Runnerin jälkeen yhä useampi peli sisälsi työkalut omien kenttien tekemiseen. Muutama ohjelmistotalo tarjosi myös työkaluja, joilla pystyi tekemään musiikkia, fontteja, spritejä tai grafiikkaa. Oman sisällön luominen oli silti pitkään rajoitettua. Työkaluilla pystyi tekemään uusia kenttiä tai muokkaamaan vanhoja, mutta pelien sisäistä logiikkaa ei juuri puukotettu.

Esimerkiksi Mr. Robot and his Robot Factory (1983) muistutti monella tapaa Lode Runneria. Siinä tasoloikittiin Loden tapaan ruuduilla, joille oli ripoteltu tasoja, tikkaita ja örkkimörkkejä. Tosin Robotin örnöt liikkuivat tarkoin määritellyillä reiteillä, siinä missä Loden vastustajat liikkuivat vapaasti.

Robotin pakissa oli yhteensä 14 työkalua, kuten liukuhihnoja, trampoliineja ja muita esineitä. Kentän rakentaminen oli yksinkertaista, mutta urakka tehtiin joystickillä ja jokainen pala piti noutaa erikseen ruudun alareunan valikosta. Niinpä siihen tuhrautui aika lailla aikaa – mutta sitähän nuorilla pelaajapojilla riitti. Toimivuuden pystyi testaamaan automatiikalla ennen pelaamista, ja jos tulos tyydytti, kentän sai tallentaa levylle.

Varhaisen kasibittisajan suurimpia hittejä oli brittiläinen Jet Set Willy (1983), joka on yhä yksi kaikkien aikojen tunnistetuimmista videopeleistä. Ovelana temppuna pelimoottori ja kenttädata oli tehty täysin itsenäisiksi kokonaisuuksiksi. Pelaajat pystyivät pian muokkaamaan pelikenttiä, sitten dataamaan koodia. Hakkerointiohjeita julkaistiin lehdissä. Pelaajat pystyivät muokkaamaan peliä ja rakentamaan omia kenttiä kaupallisilla JSW-työkaluilla. Tasoja tehtiin niin tuhottoman paljon, että fanikentistä koottiin sekä maksullisia kenttäpaketteja että pari epävirallista jatko-osaakin.

Ekat pelit, joissa pääsin tekemään omia kenttiä, olivat Kikstart 2 ja Mr. Robot Commodore 64:sella. Siihen aikaa kenttiä ei kylläkään jaettu, vaan ne tehtiin ja pelattiin heti perään. Molemmat editorit olivat yllättävän helppokäyttöisiä, hauskoja ja toimivia. Kikstart 2:sessa omat kentät mahtavilla hypyillä ja pompputorneilla ovat jääneet mieleen. Parasta ei ollut se, että sai tehdä uusia kenttiä, vaan että sai tehdä hauskan-typeriä kenttiä ja kokeiluja.

Esim. Mr. Robotissa kenttiä, jossa oli loputtomasti kuolemattomuuspillereitä, tai kenttä, jossa hahmo läpäisee sen ohjaimeen koskematta.

Tommi Äijälä

Kuuslankulla tuli läpikäytyä varmaan aika perussetti: Rally Speedway (lukematon määrä tunteja), Boulder Dash, S.E.U.C.K (joka meinasi tappaa pelientekovietin kokonaan) ja Loderunner.

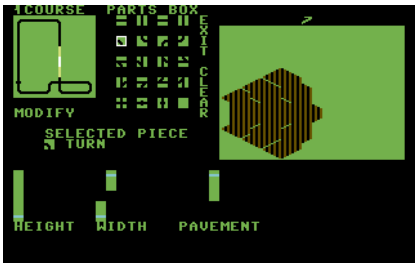
Santtu Huotilainen

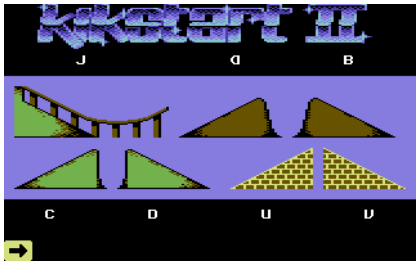
Kumi palaa

Myös ajopelirintamalla sattui ja tapahtui. Pohjimmiltaanhan hyvän ajopelin toimiva ydin löytyy pelattavuudesta. Ensin hiotaan ajaminen kuntoon, sen jälkeen tehdään radat loppuun. Etenkin vanhempien autopelien radat olivat kätevän kaksiulotteisia, joten ratasuunnittelu onnistui lätkimällä ratapaloja tasaiselle pohjalle.

Edelleen retropelibileissä suosittu Rally Speedway (1984) oli tästä nerokas esimerkki. Vauhdikkaassa kaksinpelissä autot kiitävät kilpaa ja tönivät kaveria pusikkoon. Äärimmäinen autohauskuus täydellistyi rataeditorilla, joka oli aikaisekseen hämmästyttävän käyttäjäystävällinen. Radat ja ympäristöt rakennettiin valmiista paloista, jotka valittiin ja asemoitiin näppäimistön ja tikun näppärällä yhteiskäytöllä. Pienemmät perusradat syntyivät muutamassa minuutissa.

Autoiluhulluus kiihtyi EA:n Racing Destruction Setissä (1985). Siinä ei enää tyydytty mihinkään tavanomaiseen, vaan pelaajille annettiin hurjat





työkalut ratojen rakentamiseen ja muokkaamiseen – niin maan päällä kuin vaikkapa Kuussa ja muilla taivaankappaleilla.

Painovoimaa, radan liukkautta ja muita parametrejä pystyi säätämään kattavasti, kulkupelejä riitti kuuautosta temppupyörään ja F1-kiituriin. Siisteintä oli kuitenkin ratapalojen asettelu, vääntely ja korkeuserojen säätäminen. Mikäs sen hauskempaa kuin tehdä pitkä, loiva ylämäki, jonka päässä oleva hyppyri paiskaa autot uskomattomaan ilmalentoon Kuun painovoimakentässä.

Nouskoon kolmanneksi ajopelieditoriesimerkiksi Mastertronicin oivallinen halpis Kikstart II (1987), sivusta kuvattu prätkätemppuilu. Ensimmäinen Kikstart oli ollut jo hitti, joten jatko-osaan viilattiin sekä parempi pelattavuus että kunnon editori, joka nostettiin jopa pelin kanteen ja mainoksiin. Kikstart toimikin erittäin hyvin aikansa hittibilepelinä. Omat kikkailuradat syntyivät vaivattomasti, eikä ole vaikea nähdä mistä Redlynxin Trials-sarjaa ammentaa.

Ja jos en mainitsisi Boulder Dash Construction Kittiä (1986), tämä artikkeli joutaisi jäädä kirjoittamatta. Boulder Dashit -pelit olivat mielettömiä myyntihittejä. Neliömäisistä palikoista koostuvia luolastoja oli helppo rakennella. Lopputulos puhuu puolestaan: netistä löytyy noin 2500 fanien tekemää Boulder Dash -peliä.

Lisätään Kikstart II ja Polar Pierre C64-listalle. Kuusnelosen editoripeleissä käyttöliittymät olivat hyvin lähellä varsinaista peliä. Virtaviivaista, mutta en muista, että esimerkiksi näppäimistöä olisi hyödynnetty kovin tehokkaasti, vaan hommat tehtiin pääasiassa joystickillä.

Lasse Mattila

Rally Speedway ansaitsee erityismaininnan yleisen hauskuutensa ja ajattomuutensa takia.

Chris Skog

16-bittisille tullut Mastertronicsin Sidewinder sisälsi ilmeisesti vahingossa kätketyn, pelin tekemisessä käytetyn kenttäeditorin Atari St -veriossa. Joku vääräleuka kerkesi tehdä sillä PD-levitykseen päätyneen Pakistan Pete -version, jossa lenneltiin lentävällä matolla ja ammuttiin pakistanilaisia.

Chris Skog

Muinaishistoriassa tein kenttiä Mr. Robotiin ja Rally Speedwayhin, niitä sitten tahkottiin kavereiden kanssa. Jälkimmäinen muuten tuhosi mukavasti levyn kirjoittaessaan suoraan levysektoreihin käyttämättä tiedostoja.

Jorma Oksanen

Amigan Druidiin tehtiin joskus uudet grafiikat Deluxe Paintilla. Samaten teimme Another Worldin introon useampikin eri versio musiikeista, taitaa olla vielä tallessakin. Racing Destruction Setillä tehtiin paljon kenttiä. Niissä oli vain usein se ongelma, että teimme niistä sadistisen vaikeita, kuten myös Mr.Robotin ja Boulder Dash Construction Kitin kentistä.

Aki Sivula

Suurimmalla käytöllä olivat Rally Speedwayn ja Stuntsin editorit. Yleensä tavoitteena oli tehdä rataan jokin kiero koukku, joka aiheuttaa kaverille ulosajon. Rally Speedwayssa lätkäistiin talo keskelle suoraa.

Hatih

Kikstart II oli aikansa Mario Maker ja sillä tehtiin lähinnä kaasu pohjassa ajettavia pomppukenttiä, mutta joskus vähän vakavampiakin yrityksiä. Useimmat muut muistikuvat ovat yhden ruudun tilepohjaisista sharewarepuzzlepeleistä. Ne kuitenkin innostivat jatkamaan harrastetta oikeaan peliohjelmointiin, Tile-pohjaiset pelit esimerkiksi auttoivat ymmärtämään miten kenttädataa saattoi tallentaa yksinkertaisilla taulukoilla ja parilla toisaalla määritetyllä objektilla.

Jerri

Pelien tekemistä pelinteko-ohjelmilla

Varhaisten pelieditorien ja kaupallisten pelienmuokkaustyökalujen rinnalla syntyivät myös ensimmäiset kaupalliset työkalut kokonaisten pelien tekemiseen. Ensimmäinen niistä oli EA:n Pinball Construction Set (1983). Se ei edes sisältänyt varsinaisesti peliä, vaan oli työkalupakki flippereiden tekemiseen. Pinballista kasvoi hitti, joka inspiroi EA:ia julkaisemaan myös sävellysohjelmisto Music Construction Setin, seikkailupakki Adventure Construction Setin (1984) ja jo mainitun Racing Destruction Setin.

Seikkailupelit olivat järjettömän suosittuja, ja tarinankerronnan potentiaalia lypsettiin Adventure Construction Setillä. Se mahdollisti täysimittaisten ylhäältä kuvatun roolipelin rakentamisen ja sisälsi valmiita pohjia useille seikkailutyypeille. Mikä parasta, seikkailut oli mahdollista tallentaa itsenäisenä toimiviksi peleiksi, eli luomuksia saattoi jakaa rajoituksetta kavereille. Jopa Bethesdan Todd Howard kertoo ACS:n olleen tärkeä vaikuttaja hänen uralleen roolipelien kehittäjänä.

Rapakon tällä puolella nautittiin yhä tekstipeleistä, ja Gilsoftin The Quill (1983) -tekstiseikkailutyökalut olivat sekä harrastajien että ammattimaisten pelisuunnittelijoiden – tai pikemminkin tarinankertojien – suosiossa. Pelkästään Spectrumille julkaistiin yli 450 kaupallista Quillillä tehtyä peliä. Quilliin julkaistiin grafiikkalaajennus, jonka myötä peleihin saattoi upottaa kuvia. Quill päivittyi myöhemmin Professional Adventure Writeriksi (1986), joka toimi uudemmilla kotimikroilla ja tarjoili muun muassa monipuolisemman parserin.

Toinen brittipuoti, Incentive Software, kilpaili Quillin kanssa Graphic Adventure Creatorillaan (1985). Siinä oli heti alkuunsa ymmärtäväisempi parseri ja grafiikkatuki, mutta sillä tehtiin ”vain” yli sata kaupallista seikkailua.

Eeppisin pelintekovärkki

Monimutkaisempaa pelintekemistä tarjosi Garry Kitchen's GameMaker (1985), joka sisälsi valmiit työkalut yksinkertaisten toimintapelien tekemiseen. Siinä käytettiin ovelasti valikkopohjaista ohjelmointia, jossa käyttäjä poimi haluamiaan Basic-tyyppisiä käskyjä valikosta ja määritteli niiden toiminnan joystickillä.

Vaikka ohjelmassa oli rajoituksensa, pystyi sillä tekemään varsin toimivia kokonaisuuksia. Voisinkin veikata, että sitä näpräsivät myös Sensible Softwaren pojat, jotka loivat pelintekopelien rakastetuimman ja vihatuimman ku-

ninkaan vuonna 1987. Shoot'Em-Up Construction Kit sisälsi monipuoliset työkalut niin peligrafiikan, äänten, tekoälyn kuin pelin kokonaisuudenkin rakentamiseen. Okei, myönnetään että suurin osa SEUCK-peleistä oli legendaarisen kaameaa kuraa, mutta silti: täysin ohjelmointikyvytön kämmäkin sai sillä spritet liikkeelle. Pelit pystyi tallentamaan itsenäisiksi teoksiksi, joita pystyi jakamaan eteenpäin, joten SEUCK-pelitulvaa ei pysäyttänyt mikään.

Toki SEUCK-yhteisöstä löytyi myös taitureita, joiden pelisuunnittelutaidoilla syntyi myös melkein hyviä tekeleitä. Näiden teokset pääsivät jopa lehtien kansilevyille, mikä tosin kertoo ehkä enemmän lehtien epätoivosta saada niille mitä tahansa täytettä.

Tein muutaman pelin Shoot'Em'Up Construction Kitillä, SEUCKilla. Mulla oli jopa lapsenomaisen hyvä liikeidea tehdä sillä 100 peliä ja myydä niitä ja rikastua. Olin kai ala-asteella. Doomissa tein kenttiä, joissa sai mellestää cyberdemonien ja raketinlaukaisimien kanssa. Hauskaa sen pari viikkoa.

Toni Nikkanen

Uusi vuosikymmen

1990-luku toi tee-se-itse-näpertämiseen uusia ulottuvuuksia. Atari ST:llä ja Amigalla oli jo päästy sille asteelle, että työkalut alkoivat olla hyviä, koneissa oli vähän muistiakin ja hiiriohjattu käyttöliittymä teki askartelusta astetta sujuvampaa kömpelöön joystick-veivaukseen verrattuna. Sukupolvet vaihtuivat. Kasibittiset mikrot hiipuivat vauhdilla, ja PC teki rajusti tuloaan. Peligrafiikoissa oli päästy jo VGA/SVGA-tasolle ja Creativen Sound Blaster –äänikortit toivat peleihin kunnon ääniä. PC-pelit aiheuttivat amigisteissa jo muutakin kuin naurukohtauksia.

Pelieditorejakin saatiin. Bard's Tale Construction Set (1991) syntyi kolmen ensimmisen Bard's Talen vanavedessä, ja mahdollisti omien fantasiaroolipelien tekemisen. Nimenomaan roolipelien, ei vain yksinkertaisempien seikkailujen. Mutta se oli yhä perinteinen roolipeli, kun tulevaisuus oli 3D:n.

Tämä näkyi Incentive Softwaren 3D Construction Kiteissä (1991-1992). Brittifirman Freescape-pelimoottoria käyttäneet 3D-seikkailut, kuten Driller jatko-osineen, olivat jo lyöneet isosti läpi. Kun äimistelty 3D-teknologia julkaistiin tee-se-itse-työkalupakettina, se jos mikä oli kova paukku. Nyt kaikki halukkaat pystyivät luomaan ja manipuloimaan 3D-objekteja ja ympäristöjä, rakennellen kokonaisia virtuaalimaailmoja. Massiivisen hiekkalaatikon lisäksi ohjelmisto sisälsi myös animaatiojärjestelmän ja ohjelmointikielen, joilla virtuaalimaailmaan pystyi rakentamaan toimivan peliympäristön.

Kitit olivatkin luultavasti ensimmäiset suurelle yleisölle suunnatut 3D-rakenteluympäristöt. Niitä voi pitää myös suorina esi-isinä myöhemmille netin rakentelu-virtuaalimaailmoille. Molemmilla tehdyt luomukset voitiin tallentaa itsenäisesti toimiviksi ohjelmiksi, joita voitiin jakaa kavereille.

Disneyn Stunt Island (1992) taasen vei oman sisällön tekemisen hämmentävään suuntaan, sillä se oli paitsi oivallinen lentosimulaattori, myös huikea työkalu lyhytelokuvien tekijöille. Pelin ympärillä oli täysimittainen Hollywood-stunttityökalu, jolla pelaajalle annettiin kokonainen mielikuvituksellinen saari leikkipaikaksi. Yli 50 erilaiselle lentokoneelle sai rakentaa vaikka millaisia kulisseja ja ympäristöjä, joihin ympättiin myös elokuvamainen käsikirjoitus – ja sitten ne piti vielä lentää itse. Kamerat kävivät ja tallensivat onnistuneet kohtaukset, joista pystyi lopuksi leikkaamaan elokuvan. Koska kaikki oli pelkkää dataa, Stunt Island -leffat veivät tilaa suurimmillaankin vain muutaman megatavun.

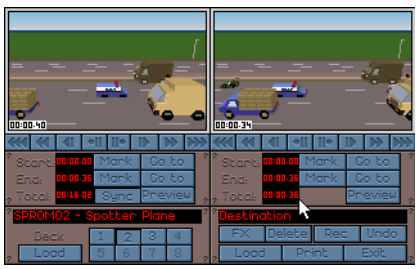
Liekö vastaavaa vapaamielistä lentopeliä tai tee-itse-elokuvaelämyksesi -peliä sen koommin tehtykään. Etenkään kun leffakohtauksia sai lentää yhtä lailla B-2-pommarilla, avaruussukkulalla tai vähän tavallisemmilla sotilas- ja siviilikoneilla.

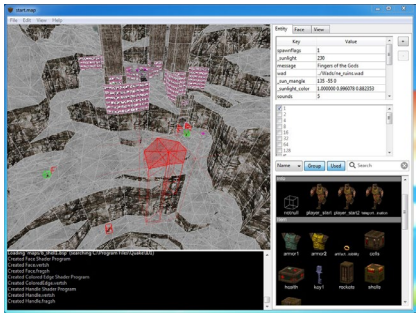
Ja mitä muut ilmassa, sitä Stunts-kuskit (1990) maassa. Elokuvia ei tosin tehty, mutta sitäkin vinkeämpiä ratoja kyllä. Hyvällä rataeditorilla syntyi valmiista palikoista nopeasti uusia ratoja, joita autokuskit vaihtelivat aktiivisesti. Kotipelaajille Stunts oli köyhän miehen versio Atarin Drivin'- ja Race Drivin' –kolikkopeleistä – ei ihan niin nätti, ei yhtä nopea, eikä se joystick nyt mikään ratti ollut. Mutta ähäs, saipa ainakin tehdä omia ratoja!

Suomalaisesta näpertelyhulluudesta vastasi Timo Kauppisen ikivihreä Slicks 'n Slide (1993), joka oli mielettömän mukaansatempaava monen pelaajan pikkuautoajelu. Pelin täysversioon sisältyi rataeditori, jolla on tehty mielettömät määrät ratoja. Niitä löytyy pikku googlauksella tuhansia, joten slicksaajille riittää yhä edelleen loputtomasti pelattavaa. Pikku vinkkauksena: Slicks-fanit löytävät hyvin samanhenkistä ajettavaa iOS:n Drift'n'Drivestä.

Command & Conquer: Red Alert puolestaan teki naksusotaeditoinnista massojen hupia, siitäkin huolimatta että EdDOS/EdWin-editorit olivat aika kämyisiä ja tarkoitettu vain moninpelikarttojen tekemiseen. Muokkaus oli hyvin rajoitettua, eikä esimerkiksi yksiköitä, rakennuksia tai valmiiden karttojen skriptausta voinut lainkaan tuunata.

Niinpä fanit rynnistivät paikalle omien editoriensa kera – ja niin syntyi kattava karttaskene myös naksusotien





megahitin ympärille. Westwood otti tästä opikseen ja sisällytti moniin myöhempiin C&C/Red Alert –sarjan peleihin kunnon työkalut, joissa ei unohdettu yksinpelikenttiä tai skriptausta.

Olin 90-luvulla vähän erilainen nuori, jolla oli kotona mäkki. Sille oli luolaräiskintä Continuum, jonka mukana tuli kenttäeditori. Sillä tuli vähän väänneltyä. Myöhemmin taisin yrittää tehdä kenttiä myös Abuseen, se tasohyppelyjen Doom. Miksi? Omaksi huviksi, oli halu olla luova. Mihinkään internetseihin en koskaan laittanut mitään. Ei tullut mieleen, että joku voisi olla kiinnostunut. Jos olisikin, niin en kuiteskaan olisi uskaltanut, kuitenkin olisivat nauraneet.

Tommi Tuura

Slicksiin tehtiin paljon ratoja, niitä sitten mätettiin keskenämme. Stuntsissa oli myös loistava editori, ajettiin kovasti kilpaa sielläkin. Doomin ja Quake1:n kenttiä tehtiin jonkin verran laneihin, mutta minua paremmat kaverit saivat ihan fameakin Q1-kentillään. Populousin paint mapillä tehtiin toisillemme haasteita, näitä ei sen kummemmin tallenneltu.

Ville Jouppi

Stuntsiin tehtiin paljon ratoja ja niitä vaihdeltiin koulussa korpuilla. Vähän erikoisempana oli Amigan Hard Drivin' 2, jossa tuli mukana buginen mutta Stuntsiin nähden vapaammat kädet antava rataeditori. Sitä väärinkäyttämällä pystyi tekemään melko huikeita ratoja. Valitettavasti itse peli oli kökkö.

Ville Ranki

Stuntsiin tehtiin ratoja ihan hulluna. Monia kenttiä joutui vähän oikomaan pärjätäkseen. Kunnes se pirun formula nousi perä edellä taivaisiin. Siinä oli se kokeilun viehätys ja näpertelyn vimma, joka yhdistyi korpuilla tapahtuvaan levikkiin.

Markus Juuti

Editointipankin räjäytti, kun pelasin läpi Eggerland-tyylisen Griel no Seihai – puzzlepelin ja vieläkin jäi sellainen fiilis, että LISÄÄ! Editoria pelissä ei ollut, joten jouduin rippaamaan grafiikat ja tekemään oman editorin. Sitten puzzleja väsättiin pirunsarvet otsassa ja voi sitä riemua, kun toinen ei millään meinannut keksiä miten kentästä pääsee läpi. Välillä tuli tietysti myös takkiin, kun huomasi toisen keksivän huomattavasti yksinkertaisemman tavan kentän läpäisemiseksi. Kenttiä pystyi kuitenkin luomaan nopeasti, ja kieron suunnitelman luomisen tuoma palkkio/pettymys oli todella koukuttava fiilis.

PC:llä eniten aikaa meni Stuntsin rataeditorin kanssa. Ratoja kiikutettiin disketillä kaverilla ja kokeiltiin kumpi vetää kovemmat ajat uudelle luomukselle. Myös The Incredible Machinen editorissa meni mukavasti aikaa.

Timo Soilamaa

Editorit vaan on niin hienoja ja tuovat lisäarvoa peliin. En todellakaan ole yrittänyt luomuksiani mihinkään, ennen kuin Trialsin kohdalla. Commodore 64:n editoreista tulee aina mieleen, että leveleitä ei loppujen lopuksi saanut oikein koskaan tallennettua mihinkään.

Sami Juntunen

Red Alert 1:n editorilla tein omaksi ilokseni kentän, jolla oli älyttömästi timantteja, jotta pystyisin mättämään vaikeaa AI:ta vastaan helpommin.

Harry-Pekka Laakso

Grand Prix Unlimitedissa oli sisäänrakennettu editori. Radat olivat tilepohjaisia, jotkin radan muodot koostuivat useammasta tilestä jotka piti laittaa paikoilleen erikseen. Tekemäni radat lienevät hukkuneet aikojen saatossa.

Stuntsin radat olivat tilepohjaisia, jotkin muodot koostuivat useammasta tilestä, mutta editori huolehti kaikkien yhteen kuuluvien tilejen asettamisesta. Tein kymmeniä erilaisia ratoja, joista pari täytti käytännössä koko pelialueen. Maastoa muistaakseni ei pystynyt muokkaamaan. Ratoja on tallessa useita kymmeniä.

Wingsin kentät tehtiin kuvatiedostoina millä tahansa kuvankäsittelyohjelmalla ja muunnettiin pelin ymmärtämään muotoon mukana tulevalla ohjelmalla. Pelissä oleva alkeellinen nestedynamiikka oli suosikkini. Muuttamalla väripalettia sopivasti veteen sai aikaan vinkeitä visuaalisia efektejä. Pelasin omia kenttiä pääasiassa pikkuveljen kanssa. Kenttiä on tallessa pari kymmentä.

Total Annihilationin editori oli varsin käyttökelpoinen, mutta kyllästyin suorakulmaisuuteen. Tein sitten oman, korkeuskäyriin perustuvan editorin. Kielenä oli C++; lähdekoodit ovat tallessa.

tdb

Command & Conquerin CCEditillä oli mahdollista muokata hahmoja ja rakennuksia. Omia rakennuksia muokattiin tietenkin tuottamaan älyttömästi sähköä, maksamaan todella vähän ja tornit ampumaan kauemmaksi. Red Alertiin tehtiin omia kenttiä kaverin kanssa. Eipä me sitten oikeastaan muuta tehtykään kuin pelattiin skirmishiä itse tehdyillä kartoilla skirmishiä tietokonetta vastaan. Aluksi tehtiin aina oma alue ja ympäröitiin se kalliolla, alue täyteen arvokkaita kristalleja. Myöhemmin opimme tekemään erilaisia jokia, meriä, saaria sekä muita kaunistuksia. Eiköhän tässäkin ollut meillä pääidea se, että pääsi vapaammin rakentamaan mahdollisimman ison tukikohdan, ilman pelkoa vihollisesta. Usein vastaavissa rakentelupeleissä sitä vain halusi rakentaa omaa paikkaansa, eikä niinkään sotia. Karttaeditorin helppokäyttöisyys on jäänyt mieleen, kun sitä kuitenkin käytettiin ala-asteikäisenä.

Dudem

Internet ja id

Uuden vuosikymmenen suurin vallankumous oli viestinnällinen. Modeemit ja BBS-purkit yleistyivät hurjaa vauhtia, ja internet kurkki kulman takana. Omien luomusten levittäminen muuttui kädestä käteen -levityksestä ja diskiswappaamisesta elektroniseksi. Modeemien nopeudet olivat nykypelaajan silmiin naurettavia, mutta pelit ja lisäkentät olivat silloin pienempiä. Nyt hyvät kentät levisivät nopeammin ja laajemmalle, maasta ja mantereelta toiselle.

Ja sitten tulivat Wolfenstein 3D (1992), Doom (1993), Duke Nukem 3D (1996) ja Quake (1996). Kaikki muuttui.

Wolfenstein 3D ei ollut ensimmäinen 3D-räiskintä, mutta ehkä genren historian merkittävin. Se oli raivokas,

vauhdikas ja asenteellinen. id Software ei hidastellut eikä pyydellyt anteeksi.

Pelien muokkaaminen oli jo yleistä, mutta Wolf3d.exe kutitti räiskintäfanien syvimpiä hermoja odottamattomalla tavalla. Peli oli ollut levityksessä vain muutaman päivän, kun ensimmäiset editorit levisivät purkeissa ja netissä. Niiden avulla kenttien salat purettiin hetkessä, ja näin pilattiin parikin tekijöiden suunnittelemaa pelaajakilpailua.

Wolfin ympärille kasvoi nopeasti mittava modaajaskene, ja fanit puukottivat Wolfin ytimiään myöten: omien kenttien, grafiikkojen ja äänien lisäksi myös sen koodillinen ulottuvuus ja jopa pelisuunnittelu avautuivat kaikille halukkaille ja osaaville.

Virittelyn valtava suosio näkyi jo seuraavana vuonna julkaistussa Doomissa. Id teki pelinsä alusta lähtien modattavaksi, ja tallensi kenttädatan erityisiin WAD-tiedostoihin (Where's All the Data?). Id ei tarjonnut muokkaustyökaluja, mutta fanit olivat pian tilanteen tasalla. WADien kautta avulla muokattiin perusteellisesti ja vieläpä luvan kanssa. Tarvitsit vain alkuperäisen Doomin, ja sitten kenttiä latailemaan.

Doom-kenttiä, pelin joksikin ihan muuksi muuttavia Total Conversioneita ja työkaluja julkaistiin niin paljon, että niistä on koostettu lukemattomia epävirallisia kaupallisia kokoelmia sekä useita virallisia ”parhaista parhaat” –koosteita. Kaiken huipuksi uusia kenttiä, haasteita ja TC:tä tehdään yhä edelleen, eikä Doom osoita minkäänlaisia kuolemisen merkkejä.

Ysärin äärimmäinen FPS-tykitys jatkui tein-sen-itse-rintamalla 1996, kun kaksi kaikkien aikojen kovinta räiskintää saapui kauppoihin. Sekä 3D Realmsin Duke Nukem 3D että id:n Quake tärisyttivät pelimaailmaa ja orgasmoivat pelaajat taivaisiin.

Vaikka molemmat pelit olivat järjettömän suosittuja ja pitkäikäisiä, vei Duke alkuun Quakea tee-se-itse-rintamalla. Pelin mukana tuli sama kenttäeditori jota tekijät itsekin käyttivät. Hyvät työkalut innoittivat porukkaa oikein kunnolla: suomalaiset tekivät Dukeen paljon hienoja kenttiä, ja taisipa MikroBitissäkin olla Duke-kentäntekokilpailu. Paljastetaan nyt sekin historiallinen salaisuus, että tarkoituksemme oli tehdä lukijoiden iloksi Bitin toimisto Duke-kentäksi, mutta valitettavasti projekti ei koskaan valmistunut.

Keräsin MikroBitin MBnet-purkkiin satoja suomalaisia Duke-kenttiä, joista monet kuvasivat muun muassa kouluja, ostoskeskuksia ja muita julkisia kohteita. Voi että olisi kukkahattusilla riittänyt niistä kauhisteltavaa. Mitä siitä, että Juhani-Ahon koulussa kävelikin avaruussikoja ja robotteja, kyllä niille olisi jokin freudilainen merkitys löydetty.

Duke oli siitä helppo, että kenttien rakenne noudatteli vielä perinteistä 2D-rakennemallia. Kenttien pykääminen ei vaatinut 3D-mallintamista, kun taas täysin 3D:nä toteutetussa Quakessa sekä haasteet että mahdollisuudet olivat jotain aivan päätöntä. id teki jopa QuakeC-ohjelmointikielen, jolla peliä oli mahdollista puukottaa syvemmältä ja perusteellisemmin kuin millään valmiilla työkaluohjelmilla.

Quaken virittely meneekin näiltä osin tämän artikkelin skaalaa pidemmälle, sillä peliin ei enää tehty pelkästään karttoja, eikä varsinkaan sen mukana tulevilla työkaluilla. Mutta samalla Quake tarjosi taitaville virittelijöille niin paljon mahdollisuuksia, että sen myötä lisäosien, tee-se-itse-laajennusten ja virittelytyökalujen markkinat räjähtivät. Ja taas pelihakkereita värvättiin pelifirmoihin.

90-luvulta muistan parhaiten Stuntsin (4D Racing) ja Action Supercross, johon kenttiä tuli väännettyä urakalla. Samoihin aikoihin demoskene ja pelinkehitys kiinnostivat, joten kokeilin vähän kaikenlaista, myös modausta, muun muassa id:n ja Valven pelejä. Suurin osa oli omaa kokeilua, mutta jotakin modeja päätyi levitykseenkin.

Tommi Äijälä

Wolfenstein 3D:hen, Stuntsiin, AUTSiin ja Slicks'n'Slideen tuli tehtyä kenttiä, Wolfiin ja Stuntsiin myös muokkasin ja spritejä. Viimeiset kentät taisin tehdä Duke 3D:hen modeemipelejä varten. Nykyään ei intoa enää riitä, kun joutuu suunnittelusoftilla tekemään ihan tarpeeksi ”kenttäeditointia”. Lienevätkö kenttien rakentelut ilmentäneet jotain varhaista hairahtuneisuutta, kun päädyin myöhemmin arkkitehdiksi. Olennaisinta koko hommassa oli, että oman luominen oli usein hauskempaa kuin valmiin pelaaminen.

Otso Helenius

Mallinsin Duke 3D:hen vanhempien talon ja lähialueiden pihapiirit. Olihan siinä ihan eri fiilis räiskiä menemään tutuissa mestoissa. Stunts oli varmaan ensikosketukseni rataeditoreihin. Siinä oli yleensä ideana tehdä rata, joka oli hankala päästä maaliin asti. Radat oli täytetty kaikenlaisilla ikävyyksillä. Hyppyreiden jälkeinen liuta betoniporsaita oli mainio hermojen tuhoaja. Hyvä jippo oli myös laittaa kovavauhtisen hypyn jälkeen tiukka mutka, ja ulkomutkaan hyppyrin suuntaisesti toinen tie, mikä vei minne sattui. Kovasta vauhdista hyppäämällä laskeutui suoraan tuonne väärälle tien pätkälle, eikä mutkaa ehtinyt edes huomata. Ah.. muistoja.

Jesse Mäkinen

Quakeen tehtiin muistaakseni kymmeniä erilaisia skinejä, muutama kenttä ja pari kökköä modia Quake C:llä.

Aki Sivula

Kaveri teki Doomiin kenttäeditorilla melko uskollisen kopion meidän lukiosta. Oli vinkeää pelata sitä samaisen lukion atk-luokassa kavereita vastaan. Nykyisin moisesta joutuisi varmaankin poliisikuulusteluihin tai vähintään koulupsykologille.

Erkki Nurmi

Tee-se-itse-artikkelin oli alkujaan määrä olla yhteensä kuusi sivua. Aihepiiri ja kommenttien määrä kuitenkin räjähtivät niin vahvasti käsiin, että artikkeli julkaistaan kahdessa osassa. Ensimmäinen osa käsittelee ajanjaksoa kenttäeditoripelien lapsuudesta noin vuoteen 2000 saakka. Lisää siis jo seuraavassa Skrollissa.

Sitä ennen lisää lukijoiden muisteloja Skrollin nettisaitilla!

Pelihistoriallinen outolintu
Metal Gear: Snake's Revenge

On olemassa Metal Gear -peli, jonka voisi luulla olevan kirottu: juonellisesti se ei kuulu pelisarjaan, se ei ilmestynyt koskaan Japanissa, eikä Hideo Kojimalla ollut mitään tekemistä sen kanssa. Konamin ja Kojiman välisen vihanpidon ensiaskeleet taidettiin ottaa jo tuolloin, kun Kojimalle "unohdettiin" kertoa koko pelin olemassaolosta.

Teksti: Pasi Huttunen Kuvat: Pasi Huttunen, Jukka O. Kauppinen, Konami, Janne Katajamäki

Peli on 25 vuotta sitten NES:lle ilmestynyt Snake's Revenge. Peli sai Kojimalta arvion "Somewhat a crappy game" ja toimi kimmokkeena upean upean Metal Gear 2:n suunnittelulle.

Mutta mikä Snake's Revenge on? Miksi sillä on niin hämärä historia? Miksi siitä ei pidetty? Ja miksi minä en yhdy kriittisten kuoroon, vaan muistelen sitä kaihoisasti? Sukellus historiaan tarjoaa vastaukset näihin kysymyksiin.

Metal Gearin synty

Metal Gearin syntytarina on erikoinen. Kojiman legendaarisen hiippailutoimintapeligenren synnyttänyt teos oli alun perin kuin itänaapurista tuttu suristin: se ei sisältänyt hiippailua, eikä se ollut Kojiman.

Tapahtumaketju alkoi vuodesta 1983, jolloin Microsoft lanseerasi MSX-arkkitehtuuristandardin. Se oli kotitietokoneille sama kuin VHS oli videonauhureille: eri valmistajien MSX-standardia noudattavilla tietokoneilla pystyi pelaamaan samoja pelejä. Standardi otettiin ilolla vastaan muun muassa Japanissa, Suomessa ja isossa osassa Eurooppaa, kun taas USA:ssa se ei lyönyt läpi.

Konami liittyi MSX:n tukijaksi jo varhain. Eräs työn alla olevista peleistä oli vauhdikas toimintapeli, jossa ruutu pursuaa ammuksia ja lyijynamia kaipaavia vihollisia. Toteutus tyssäsi, kun silloinen MSX-standardin uusin versio, MSX2, ei pystynyt käsittelemään ruudulla yhtä aikaa tarvittavaa määrää vihollisia ja ammuksia.

Pelin kehitysvastuu annettiin Kojimalle, joka alkoi miettiä miten tekniikan rajoitukset voisi kiertää. Ratkaisu oli selvä: vaihdetaan toiminta hiippailuun ja pakoiluun, osin The Great Escape -pelin innoittamana. Tällä muutoksella ruudulla yhtä aikaa olevien vihollisten määrää saatiin rajoitettua pelimaailmaan sopivalla tavalla. Peli sai nimekseen *Intruder*.

Kojimalla oli käsissään mestariteos, joka olisi muuttava pelimaailmaa. Yksinkertaisen toiminnan sijaan peli

tarjosi muun muassa kevyitä roolipelielementtejä vankeja pelastamalla nousevan sotilasarvoasteikon myötä, radiokeskustelua, hiippailua, käytettäviä esineitä ja vaikka mitä. Vartijoillakin oli silmät ja korvat, ja he tekivät hälytyksen outoja havaitessaan. Vaikka nämä ominaisuudet ovat nykyään peruskaurapuuroa, siihen aikaan kyse oli uraa uurtavasta monimutkaisuudesta ja realismista. Peli ilmestyi lopulta vuonna 1987 ja sai nimen *Metal Gear*.

Ensimmäinen petos

Kojima sai tuta Konamin ihanuuden ensimmäistä kertaa jo vuonna 1987. Konami näet halusi Metal Gearin myös Amerikkaan, mutta koska MSX2 ei ollut rapakon toisella puolen suosittu tietokone, haluttiin peli julkaista MSX2:n sijaan Nintendon NES-konsolilla. Kerrottiinko Kojimalle tästä? No ei.

Seurasi työkoneiden arvonta. Toteutusvastuu annettiin yhdelle Tokion Konamin osastoista, eikä tietenkään sille, jolla Kojima työskenteli. Käännösprojekti tehtiin kaiken lisäksi juosten pissien. Osastolle lähetettiin pelin MSX2-version lähdekoodi, annettiin aikaa vaivaiset kolme kuukautta ja saatesanoiksi viesti: tehkää pelistä mahdollisimman erilainen kuin Kojiman.

Tiimi teki mitä käskettiin. Peli julkaistiin Japanissa joulukuussa 1987 Famicom-konsolille, siis sikäläiselle NESille. Lokalisoitu NES-versio Pohjois-Amerikan markkinoille syntyi kesäkuussa 1988; Eurooppaan ja Australiaan peli saapui 1989.

Peli sai varsin nihkeän vastaanoton. Pian huomattiin, että se oli vain vesitetty versio alkuperäisestä MSX2:n pelistä. Pelistä puuttui muun muassa ammusten ja ruokatarvikkeiden hankkiminen, joka siis tapahtui vartijoita nyrkillä muksien. Niin hyvä peli kuin alkuperäinen Metal Gear olikin, Famicom/NES-version tekniset ongelmat ja konsolin rajoitteista johtuvat sisällölliset laimennukset suututtivat pelaajat ja arvostelijat.

Käärmeen kosto eli toinen petos

Tässä vaiheessa olisi voinut luulla Konamin tulevan hattu kourassa Kojiman luo, mutta ehei. Anteeksipyynnön sijaan puukosta tuli selkään toistamiseen.

Kun alkuperäistä Metal Gearia myytiin kuin leipää eikä Kojimaa kiinnostanut jatko-osan tekeminen, päätti Konami ottaa ohjat omiin käsiinsä. Taas.

Jatko-osapullat NESin Metal Gearille laitettiinkin pikimmiten uunin. Työ annettiin samalle osastolle, joka teki alkuperäisen käännöksen. Kojimalle ei edes kerrottu jatko-osan olevan työn alla.

Reilukerhon kokoontumisella oli kuitenkin lopulta onnellinen loppu. Kojima nimittäin törmäsi paikallisjunassa Snake's Revengen kehittäjään, joka pyysi häntä tekemään oikean jatko-osan. Kojimaa sieppasi sen verran tuuheasti, että aidon Metal Gear 2:n suunnitelma oli raapusteltu nenäliinaan jo ennen junamatkan päättymistä.

Metal Gear 1.5 ja 2.0

Snake's Revenge julkaistiin vain NES-konsolille Pohjois-Amerikassa huhtikuussa 1990 ja Euroopassa 1992.

Vaikka Snake's Revenge oli edeltäjäänsä – Metal Gearin Nintendo-vesitystä – niin paljon isompi ja parempi, se ei yltänyt ikiaikaiseen klassikkostatukseen. Osasyy oli, että kun Kojiman virallinen jatko-osa Metal Gear 2: Solid Snake ilmestyi heinäkuussa 1990 MSX2:lle, tuntui Snake's Revenge sen rinnalla Metal Gear 1.5:lta.

Kojiman MG2:ta pidetään yhtenä kaikkien aikojan parhaista 8-bittisistä peleistä. Jos MG1 mullisti pelimaailman, MG2 oli kaikessa suurempi, parempi ja hienompi kuin edeltäjänsä. Koska peli ilmestyi vain reilu kuukausi epävirallisen Snake's Revengen jälkeen, katosi Snake's Revenge pelikartalta yhtä nopeasti kuin kolikot pesukoneeseen.

Nyt jo kaksi selkäänpuukotusta kärsinyt Kojima maksoi perunat perunoina ja päätti, ettei hän huomioi tulevissa Metal Gear -peleissä Snake's Revengen juonta millään tavalla. Tämän seurauksena siitä tuli Metal Gear -perheen hylkiölapsi, jota päin syljetään kaduilla.

Jokainen hyvä peli on artikkelin arvoinen

Miksi kirjoittaa artikkeli pelistä, joka sikisi pahuuden työstä ja kalpenee aidon jatko-osan rinnalla? No siksi, että Snake's Revenge on erinomainen peli. Sen ansioita ei vähennä se, että Metal Gear 2 on vielä parempi.

Nykyajan emulaattoriaikakaudella on helppo mehustella pelisarjoja ja painia, mikä onkaan eri pelien paras versio ja millä laitteella. Usein unohtuu se, että flanellipaitojen vuosikymmenellä NESin omistajille oli tarjolla vain kaksi Metal Gear -peliä: umpisurkeasti versioitu Metal Gear 1 ja metallinkova jatko-osa Snake's Revenge.

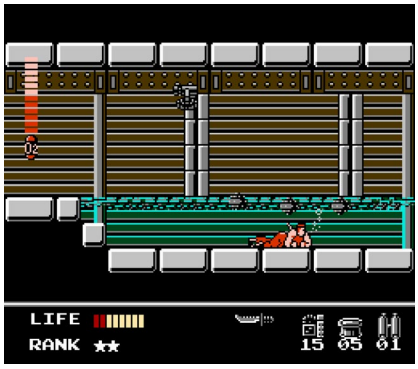
Ensikosketus

Unohdetaan hetkeksi asian tarkastelu jälkiviisaasta pelihistorianäkökulmasta ja hypätään nappiverkkarit kahisten aikamatkalle katsomaan pienen nelosluokkalaisen pojan elämää maaseudulla. Pikkukylässä, jossa räkänokkaisena asuin, ei pahemmin ollut pelikonsoleita saati peli-ikäistä nuorisoa. Poikkeuksena oli kaverini, joka oli onnistunut kärttämään itselleen kuponkiuutisista tilatun Nintendo-kloonin. Sellaisen,

jolla toimi joka kolmas kaupan peli.

Makasin sohvalla napa täynnä kinkkua, kun lankapuhelin pirisi. Pukki oli kuulemma tuonut sellaisen pelin, että oksat pois ja osa juuristakin. Puhelun jälkeen mietin hetken uskoako. Siihen saakka olimme pelanneet pääosin vain 50 in 1 -pelejä, ja kaikkein monimutkaisin näkemäni peli oli Super Mario 3. Silti pidin jo silloin tarjolla olevia pelejä liian yksinkertaisina. Kaipasin haastetta, juonta, jotain suurta ja kaunista.

Sivakat jalkaan ja pakkaseen hiihtämään sudenjätöksiä hangilla väistellen. Matkalla mielikuvitukseni jo liihotti: Voisikohan tässä olla monimutkaisuutta? Juonta? Realistinen maailma? Kehittyvä hahmo?



Mitä tässä pitäisi tehdä?

Loputtomien sauvantyöntöjen jälkeen istuin kaverini olohuoneen lattialla, jäisiä sormiani lämmittäen ja ruutua epäuskoisena tuijottaen. Koska kolmas kotimainen ei siinä iässä vielä oikein sujunut, ei meillä ollut hajuakaan mitä pelissä pitäisi tehdä. Juostiin viidakossa, ammuskeltiin vihollisia, laukaistiin hälytyksiä ja sitten saatiin kipeästi sipuliin vihollislauman hyökätessä paikalle.

Hölmöltä ja vaikealta toimintapelilltä alussa tuntunut kostokärmes alkoi kuolema kuolemalta avautua. Ahaa, kun vartijan niistää luuvitosella, tipahtaa siltä tarvikkeita. Eikä senkkaa kannata nenästään kaivaa turhaan, sillä viekkaudella yhteenotot voi väistää kokonaan. Eteneminen kuitenkin pysähtyi lopulta kielitaidon puutteeseen sekä Kaunareiden alkamiseen. Pojat ulos hankeen leikkimään!

Sissisulatejuustoa läpipeluuohjeilla

”Peli on niin monimutkainen, että vaatii läpipeluuohjeet”, ihastelimme. Kaikki siihen saakka pelaamani pelit olivat perustuneet yksinkertaisen pistemaksimin jahtaamiseen tai pelikenttien läpäisyyn.

Immersiivinen pelimaailma auttoi pelihahmoon samaistumisessa. Saatoin melkein haistaa märän viidakon ja sissisulatejuuston tuoksun, tuntea maiharien puristavan varpaitani ja selässä painavan repun sekä vastuun. Kyykin puskassa vartijan toimia seuraten ja sopivan hetken tullessa tarjoilin rystysämpylää kaikilla täytteillä suoraan takaraivoon. Pistoolikin poltteli taskussa, mutta ilman äänenvaimenninta siihen ei kannattanut koskea.

Kun kiusalliset pari kuukautta myöhemmin läpipeluuohjeet lopulta ilmestyivät, pyöriminen aloitusalueella loppui

Ensimmäisistä Metal Geareista on tehty lukuisia versioita ja väännöksiä. Tässä lista eri versioista.

Metal Gear (MSX2)
Kojiman: Kyllä.
Englanninkielinen versio on käännetty amatöörimäisesti, ja tilanpuutteen takia siitä puuttuu iso osa tekstisisältöä.
Nykysaatavuus: Wii Virtual Console (Japani).
Remake: Kyllä, löytyy pelistä Metal Gear Solid 3: Subsistence sekä kokoelmilta Metal Gear Solid HD Collection ja Metal Gear Solid: The Legacy Collection. Remake-versio on etenkin käännösten osalta paranneltu versio alkuperäisestä.

Metal Gear (Famicom / NES)
Kojiman: Ei.
Amatöörikäännös. Vesitetty versio MSX2-versiosta.
Nykysaatavuus: Japaninkielinen versio sisältyy peliin: Metal Gear Solid: The Twin Snakes, Premium Package Edition.

Metal Gear (C64, MS-DOS)
Kojiman: Ei.
Sekä C64- että MS-DOS-käännökset pohjautuvat vesitettyyn NES-versioon, ei alkuperäiseen MSX2-versioon.

Snake's Revenge (NES)
Kojiman: Ei.
Jatko-osa Metal Gearin NES-versiolle, julkaistu vain englanniksi. Mukana tuleva ohjekirja on täynnä huonoja vitsejä, eikä juonikuvaus liity itse peliin. Ei virallinen osa Metal Gear -juonijatkumoa.

Metal Gear 2: Solid Snake (MSX2)
Kojiman: Kyllä.
Virallinen jatko-osa Kojiman MG1:lle. Arvostetaan yhtenä parhaista 8-bittisistä peleistä.
Nykysaatavuus: Wii Virtual Console (Japani).
Remake: Kyllä, löytyy samoilta levyiltä kuin Metal Gear (MSX2). Remake-versio on hieman paranneltu/muutettu versio alkuperäisestä.

”No kannattaako tätä pelata?”

Jos et ole koskaan pelannut MSX2:n Metal Gear 2:ta, niin kyllä! Jos olet, testaa silti. Peli on kestänyt aikaa erinomaisesti, eikä nostalgiakakkuloita tarvitse nenälle painamaan. Vaikeustaso on välillä armoton, mutta tallennusmetodi estää pysyvän kaljuuntumisen.

”No miten voin pelata tätä?”

Snake's Revenge on ainoa Solid-sarjaa edeltävä Metal Gear, josta ei ole tehty nykykonsoleilla toimivaa versiota. Ainoa tapa päästä käsiksi tähän herkkupalaan on käyttää Nintendo-emulaattoria. Vaihtoehtoja on monia, mutta yksi hyväksi todettu emulaattori on FCEUX.

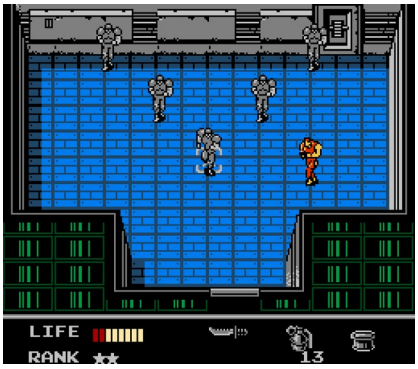
Emulaattori löytyy osoitteesta www.fceux.com. Download-linkin takaa löytyy kaksi eri versiota: win32-binary Windowsille ja SDL-versio Linuxille sekä Mac OS X:lle. Windows-version asennuksessa ei ole kummallisuuksia, mutta SDL-versio vaatii hieman viitseliäisyyttä, koska ohjelman saattaa joutua kääntämään lähdekoodista. Se löytyy kuitenkin myös monen Linux-jakelun paketinhallinnasta.

Versiosta riippumatta FCEUX:n käyttö on varsin suoraviivaista. Muista säätää ensin näyttöasetukset kuntoon kasvattamalla ruudun X- ja Y-parametreja samassa suhteessa. Halutessasi voit käyttää pehmennysfiltteriä, mutta peli näyttää varsin hyvältä ilmankin. Emulaattorin lisäksi tarvitset toki vielä pelin ROM-tiedoston.

Paras tapa nauttia peli autenttisesti on olla käyttämättä emulaattorin tallennusominaisuutta, paitsi silloin kun pelaamisen lopettaa hetkeksi. ”Continue” vie sinut tarpeeksi lähelle pistettä, johon jäit, mutta peliin rakennettu salasanan tallennustapa sen sijaan saattaa heittää tuntikausia taaksepäin.

ja upposimme peliin hikisiä kainaloitamme myöten. Läpipeluuohjeet eivät spoilanneet liikaa, vaan ohjasivat päätavoitteissa sopivan niukkasanaisesti.

Ahaa, pitää muistaa hakea avainkortti 1. Ahaa, pitää soittaa radiopuhelimella kotiin, jotta sieltä hoidetaan minut vihollistukikohtaan sisään.



Paras. Peli. Ikinä.

Näin immersiivistä pelimaailmaa en ollut koskaan kokenut. Kehittyvä hahmo! Näkevät ja kuulevat vartijat! Hiippailua! Väistelyä! Viekkautta! Juonta!

Pääsin nauttimaan kymmenistä eri aseista, lukuisista jännittävistä miljöistä, petoksista, radiopuhelimesta loppuvista pattereista, pelimaailman pulmia ratkaisevista tavaroista ja perspiitä karmivista pomotaisteluista, joissa kaikki siihen asti kerätyt taidot ja varusteet piti laittaa todelliseen hyötykäyttöön. Välillä jopa seikkailtiin sivuttaissuuntaisessa 2D-maailmassa.

Pelatessamme koimme kaikki tunteet, mitä maailmasta löytyy. Välillä naurettiin pahisten vihapuheille. Sitten viskottiin ohjaimia, kun saatiin sadatta kertaa silmäämme samalta pomoviholliselta. Lopulta onnistuminen kuivasi kyyneleet räkänokistamme. Jännitimme juonen kaarta, keräilimme lattialta suusta yllätysten jälkeen tippunutta purkkaa, ja pelin loputtua molempien silmästä löytyi roska. Etenkin kun peli ei lopu kaikille hahmoille onnellisesti.

Jokaisella on luultavasti oma messiaspelinsä. Joillekin se on Dungeon Master, joillekin Final Fantasy 7, minulle ja kaverilleni se oli Snake's Revenge. Sen lisäksi, että se on yksi Nintendon parhaita pelejä, se on minulle kuin ensirakkaus. Vaikka muitakin ihanuuksia tulee elämän polulla vastaan, joskus iltaisin sitä miettii mitä sille ihan ekalle rakkaudelle kuuluu. Hyvää kuuluu, ajan ruosteinen hammas kun ei ole nakertanut peliä pätkääkään.

Mitkä peleistä kannattaa pelata ja missä järjestyksessä?

1) Metal Gear -remake
2) Snake's Revenge
3) Metal Gear 2 -remake

Metal Gear -sarjan historiaa

Teksti: Jukka O. Kauppinen

Tästä se alkoi. Ensimmäisestä Metal Gearista.

Vuonna 1987 käynnistyneessä pelisarjassa on nykyään jo kolmisenkymmentä julkaisua, joskin osa näistä osa on uudelleenjulkaisuja ja remiksauksia. Hideo Kojiman luotsaama pelisarja nousi etenkin Snake Eaterin aikoihin omituisiin sfääreihin, sillä Kojiman taiteellinen luonne loitsi teoksista enemmänkin interaktiivisia elokuvia kuin pelkkiä pelejä. Pahimmillaan Metal Geareissa katsottiin enemmän välivideoita kuin pelattiin.

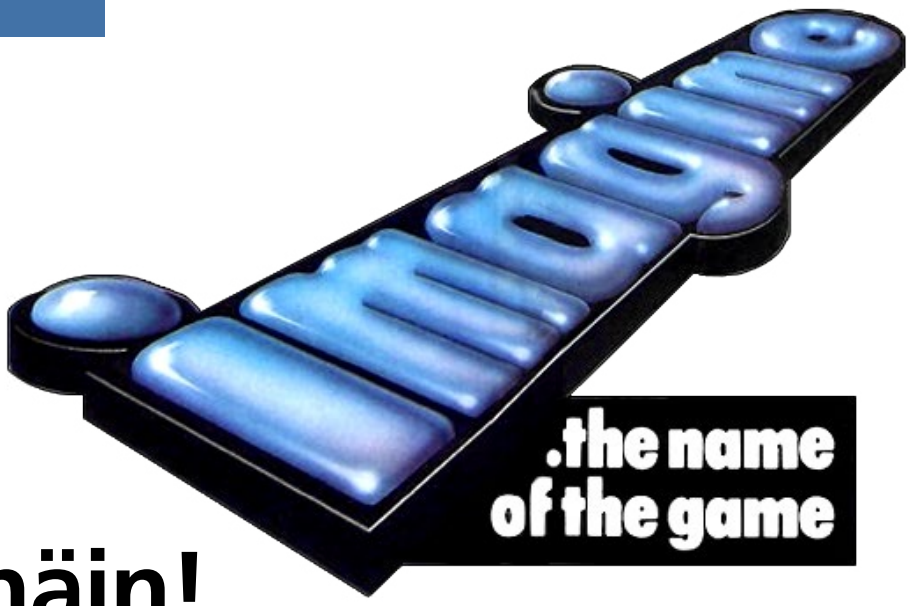
Kojiman ja Metal Gearin tarina päättyi vuonna 2015 dramaattiseen välirikkoon, kun pelisarjan omistava Konami hankkiutui hiljalleen eroon pelimegastarastaan ja lopulta lähetti tämän "lomalle". Konami myös siivosi Kojima Productions -studion nimen pelien kansista, minkä jälkeen koko studio suljettiin.

Peliyhtiö ilmoitti kuitenkin jatkavansa Metal Gear -pelien kehittämistä. Aika näyttää, mitä suunnitelmista tulee, sillä Metal Gear V: The Phantom Pain -pelistä huomaa selvästi, missä kohtaa Kojima heitettiin yli laidan. Peli harhautuu loppua kohden mitäänsanomattomaksi koheltamiseksi, jossa on vain liimailtu kasaan yksittäisiä palasia ilman minkäänlaista näkemystä kokonaisuudesta. Hauskaa sekin toki on, ja Phantom Pain on tyylikäs lopetus Kojiman ja Metal Gearin yhteiselle polulle.

Oheisella listalla on Metal Gear -saagan pääpelit ja kiinnostavia poimintoja muista, mutta ei uudelleenjulkaisuja tai remixejä.

Metal Gear	1987 – MSX2
Snake's Revenge	1990 – NES
Metal Gear 2: Solid Snake	1990 – MSX2
Metal Gear Solid	1998 – PlayStation, PC
Metal Gear Solid: Integral	1999 – PlayStation, PC
Metal Gear: Ghost Babel	2000 – Game Boy Color
Metal Gear Solid 2: Sons of Liberty	2001 – PlayStation 2
Metal Gear Solid: The Twin Snakes	2004 – GameCube
Metal Gear Solid 3: Snake Eater	2004 – PlayStation 2
Metal Gear Acid	2004 – PSP
Metal Gear Solid: Portable Ops	2006 – PSP
Metal Gear Solid 4: Guns of the Patriots	2008 – PlayStation 3
Metal Gear Online	2008 – PlayStation 3
Metal Gear Solid Touch	2009 – mobiili
Metal Gear Solid: Peace Walker	2010 – PSP, PlayStation 3, Xbox 360
Metal Gear Arcade	2010 – kolikkopeli
Metal Gear Rising: Revengeance	2013 – PC, PlayStation 3, Xbox 360
Metal Gear Solid V: Ground Zeroes	2014 – PC, PlayStation 3, PlayStation 4, Xbox 360, Xbox One
Metal Gear Solid V: The Phantom Pain	2015 – PC, PlayStation 4, Xbox One

Metal Gear -pelien päähenkilön kehittyminen vuosien ja vuosikymmenten mittaan.



Ei näin!
Liian iso keikka

Kunnianhimo on menestyksen edellytys. Voidakseen kasvaa on yrityksen otettava riskejä ja ajoittain lyötävä kaikki pelimerkit pöytään kerralla. Raja rohkeuden ja suuruudenhulluuden välillä on kuitenkin usein veteen piirretty.

Teksti: Mikko Heinonen
Kuvat: Manu Pärssinen, Mikko Heinonen
Lehtiskannaukset: RetroPDFs-projekti

Euroopassa pelibisneksen edelläkävijämaa oli Iso-Britannia. Siellä syntyi jo 1980-luvun alussa monia ohjelmistojulkaisijoita, joista parhaat ponnahtivat hetkessä miljoonien puntien vuosimyynteihin. Eräs keihäänkärjistä oli vuonna 1982 perustettu Imagine Software, joka tehtaili hittejä monelle eri alustalle. Se oli hyvin edustettuna, kun vuoden 1983 Golden Joystick -palkintoja jaettiin. Myös omistajat oppivat nopeasti makean elämän alkeet: pyllyn alle löytyi iso BMW ja vapaa-aikana sponsoroitiin vaikkapa ratamoottoripyöräilyä. Yhtiöstä ja sen pelintekijöistä tulikin keltaisen lehdistön lemmikkejä, joiden edesottamuksista raportoitiin kuin isoistakin julkkiksista.

Vuoden 1984 joulumarkkinoille Imagine aikoi tuoda jotakin aivan uutta: megapelit. *Bandersnatchin* (Sinclair Spectrum) ja *Psyclapsen* (C64) piti olla aivan muuta kuin kilpailijoiden tarjoomukset. Pelejä tekemään oli haalittu yhtiön paras osaaminen, ja niitä rummutettiin lehdissä jättikampanjalla jo ennen kuin riviäkään oli koodattu. Itse asiassa etenkin Bandersnatch oli niin suuri ja monimutkainen peli, että se vaati Sinclair Spectrumiin myös lisälaitteistoa ääntä ja kuvaa varten. Normaalin vajaan kymmenen punnan sijaan kasetilla ja rautalaajennuksella olisi hintaa melkein neljäkymmentä puntaa.

Artisti maksaa

Imaginen johtoportaaseen kuulunut Bruce Everiss kertoi myöhemmin haastattelussa, että yhtiön johtamiskulttuuriin kuuluivat kurin puute ja hallitsematon luomisvoima. Nykyisin sitä varmaankin kutsuttaisiin startuppöhinäksi. Ohjelmoijat saivat innovoida täysin rajoituksetta, eikä heidän luovuuttaan haluttu kahlita millään tavoin.

Tämä aiheutti osaltaan sen, ettei työnjohdon käsitettä suuremmin tunnettu, joten työt eivät aina edenneet suunnitellusti. Kun projektit päättyivät, ei väkeä yleensä irtisanottu, vaan lopulta ”töissä” kävi porukkaa, jolla ei ollut oikeastaan mitään tekemistä.

Bandersnatchin laitteistolaajennus sai alkunsa siitä, että piratismin torjumiseksi haluttiin luoda Spectrumin käyttäjäporttiin kytkettävä palikka, jota ilman peli ei käynnistyisi. Ensimmäisissä suunnitelmissa se sisälsi pari vastusta. Kuuluisa mopo karkasi kuitenkin käsistä ja lopulta dongleen visioitiin lisämuistia, matematiikkaprosessoria sun muuta. Pelikasetin kylkeen taas oltiin paketoimassa jopa soundtrackin sisältävää LP-levyä. Kilpavarustelu paisutti Bandersnatchin nopeasti tasolle, josta oikeastaan kenelläkään koko bisneksessä ei ollut aiempaa kokemusta. Kun muuta näytettävää ei ollut, lehdissä mainostettiin muun muassa sitä, kuinka monta virvoitusjuomatölkkiä ohjelmoijat ovat juoneet.

Onnettomuus tallentui videolle

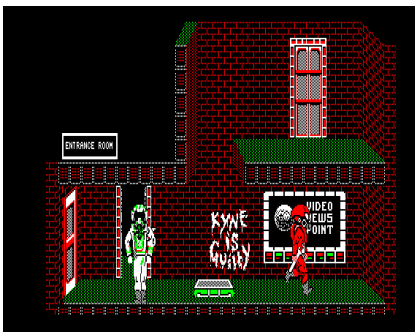
Kunnianhimon puutteesta Imaginea ei ainakaan voinut syyttää. Rautalaajennuksen valmistaminen olisi maksanut liki tuplasti firman edellisvuoden liikevoiton verran, eikä vuodesta 1984 ollut tulossa lainkaan edellisen kaltaista. Itse asiassa huhuja firman talouden huonosta tilasta oli kuulunut jo loppuvuodesta 1983. Pelimyynti takelteli pahasti alkukesän kuukausina, jolloin rahaa olisi kipeästi kaivattu tuotteiden viimeistelyyn joulumarkkinoita varten. Ulkopuolisilta sijoittajilta ei löytynyt uskoa uusimpaan jättiprojektiin. Lopulta oltiin tilanteessa, jossa kukaan ei enää suostunut myymään Imaginelle edes C-kasetteja pelejä varten.

Kuinka sattuikin, BBC:n kuvausryhmä oli juuri parhaillaan tekemässä dokumenttia Ison-Britannian peliteollisuudesta, ja Imagine oli sen keskiössä. ”Commercial Breaks” -nimiseen dokumenttiin tallentuivat Imaginen johtajien kasvavat huolet ja talouspäällikön synkät neuvot siitä, miten mihinkään kustannuksiin ei saa enää sitoutua.

Mainos Computer & Video Games -lehdessä, helmikuu 1984.

Mainos Computer & Video Games -lehdessä, maaliskuu 1984.

Mainos Crash-lehdessä, huhtikuu 1984.



Brataccas (1985, Amiga-versio).

Huipentumana nähdään lähes absurdi hetki, jolloin ulosottomies sulkee toimiston oven lounaalta palaavan henkilökunnan edestä. Pätkä löytyy YouTubesta valitettavan heikkolaatuisena, mutta on ehdottomasti katsomisen arvoinen.

Kuolema ei ole lopullinen

Pikakiihdytystä seurasi äkkipysähdys, ja Imagine Software putosi kiertoradalta lähes yhtä nopeasti kuin oli sinne noussutkin. Toiminimen osti konkurssipesältä kilpailija Ocean Software, joka palkkasi myös osan talon kyvykkäimmistä ohjelmoijista. Imagine jatkoi Oceanin alaisena studiona, ja sen nimen saattoi nähdä emoyhtiön julkaisemien pelien kannessa vielä useamman vuoden ajan.

Kahdesta ”megapelistä” Psyclapse oli täydellistä vaporwarea: se oli olemassa lähinnä mainosmiesten korulauseissa. Bandersnatch sen sijaan oli olemassa – erinomaisen keskeneräisenä, mutta kuitenkin. Imaginen rippeistä syntynyt Finchspeed-niminen yhtiö osti sen oikeudet ja alkoi työstää peliä valmiiksi, tällä kertaa tehokkaammalle laitteelle ja ilman rautalaajennusta. Valitettavasti kohdealustaksi valittiin Sinclair QL, joka ei milloinkaan saavuttanut edeltäjänsä kaltaista suosiota. Peliä ei ehditty julkaista QL:llä, sillä myös uusi kehittäjä meni konkurssiin.

Kuten pelialalla on tapana, Finchspeedin raunioista syntyi jälleen uusi studio: Psygnosis. Sen ensimmäinen peli oli nimeltään *Brataccas*, joka julkaistiin Amigalle, Atari ST:lle ja Macintoshille. Ei tarvitse tehdä kummoistakaan uskonhyppyä arvatakseen, mistä pelin runko oli kotoisin. Epäselvää on kuitenkin se, paljonko Brataccas lopulta sisälsi jo Imaginen aikana syntynyttä pelilogiikkaa.

Hieman erikoinen, tarkkuusgrafiikkaa ja hiiriohjausta käyttänyt Brataccas ei saavuttanut kovin suurta suosiota, mutta itse Psygnosis kohosi muutamassa vuodessa maineeseen laadukkailla julkaisuillaan. Shadow of the Beast näytti, mihin Amiga pystyy – DMA Designin kehittämä Lemmings taas tuhosi tuottavuuden kaikkialla maailmassa. Lopulta Sony osti koko yrityksen vuonna 1993 ja nimesi sen SCE Studio Liverpooliksi 2000-luvun taitteessa. Studio Liverpool suljettiin vuonna 2012, mutta Sony toimii paikkakunnalla edelleen.

Myös Psyclapse sai lopulta oman sivujuonteensa historiassa. Psygnosis nimittäin valitsi sen aputoiminimekseen, jolla julkaistiin muutamia pelejä 1990-luvun alussa.

Go big or go home

Imaginen kaatumisen syistä on liikkeellä monta teoriaa, hieman riippuen siitä, keneltä asiaa kysytään. Etenkin Bruce Everiss on yhä tiukasti sitä mieltä, että yrityksen kaatoi etupäässä pelien kopiointi. Osalla pelejä myyvistä liikkeistä oli niin löyhä palautuskäytäntö, että pelin saattoi ostaa, käydä kotona kopioimassa ja palauttaa sitten myymälään. Imagine sai liikkeistä palautuksina tuhansia pelejä, joissa ei ollut mitään vikaa. Megapelit syntyivät vastaamaan tähän tarpeeseen: kun puolet pelistä olisi fyysisellä palikalla, pelkän kasetin kopioimisella ei voittaisi mitään.

Toinen näkemys asiaan on, että Imaginen taloudenpito ja johtaminen olivat aidosti pielessä. Muutaman hitin jälkeen yritykseen iski vauhtisokeus, jossa kaikki ratkaisut nähtiin hyvinä, töihin otettiin kaikki kynnelle kykenevät eikä talousosasto saanut pidettyä kuluja kurissa. Megapelit olivat tämän huipentuma, amok-juoksu, jonka lopputulos oli oikeastaan jo tiedossa. Historia tuntee lukemattomia vastaavia tapauksia eri aloilta.

Oli totuus sitten kumpi tahansa tai jokin niiden yhdistelmä, Imagine ei ollut yhdentekevä yhtiö. Paitsi, että firman kaatuminen vain reilun vuoden mittaisen tähdenlennon päätteeksi tallentui jälkipolville kiintoisaan dokumenttiin, se myös synnytti lopulta Psygnosisin, josta tuli eräs Euroopan maineikkaimpia pelialan toimijoita. Joskus on ensin epäonnistuttava kunnolla, että voi onnistua.

PR2
Itella Posti Oy
Itella Green

```
; colors
move.l #$01800fd3,(a6)+ ; color 0
move.l #$01820832,(a6)+ ; color 1
move.l #$0184036b,(a6)+ ; color 2
move.l #$01860667,(a6)+ ; color 3
move.l #$01880f53,(a6)+ ; color 4
move.l #$018a07ad,(a6)+ ; color 5
move.l #$018c0000,(a6)+ ; color 6
move.l #$018e0cef,(a6)+ ; color 7

move.l #32,d0 ; Number of iterations
move.l #$07,d1 ; Current row wait
move.l #sin32_15,a0 ; Sine base
move.l frame,d2 ; Current sine
scrollrows:
        ; Wait for correct offset row
        move.w d1,(a6)+
        move.w #$fffe,(a6)+
        ; Fetch sine from table
        move.l d2,d3
        and.l #$1f,d3
        move.b (a0,d3),d4
        ; Transform sine to horizontal offset value
        move.l d4,d5
        lsl.l #4,d4
        add.l d4,d5
        ; Add horizontal offset to copperlist
        move.w #$0102,(a6)+
        move.w d5,(a6)+
        ; Proceed to next row that we want to offset
        add.l #$500,d1
        ; Move to next sine position for next offset row
        addq.w #1,d2
        subq.w #1,d0
        bne scrollrows

        ; end of copperlist
        move.l #$fffffffe,(a6)+
```

Reaktor
reaktor.com/careers

6 430057 375747